UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

Content-Based Image Retrieval of Museum Images

by

Mohammad Faizal Ahmad Fauzi

A thesis submitted for the degree of
Doctor of Philosophy

August 2004

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Content-Based Image Retrieval of Museum Images

by Mohammad Faizal Ahmad Fauzi

Content-based image retrieval (CBIR) is becoming more and more important with the advance of multimedia and imaging technology. Among many retrieval features associated with CBIR, texture retrieval is one of the most difficult. This is mainly because no satisfactory quantitative definition of texture exists at this time, and also because of the complex nature of the texture itself. Another difficult problem in CBIR is *query by low-quality images*, which means attempts to retrieve images using a poor quality image as a query. Not many content-based retrieval systems have addressed the problem of *query by low-quality images*.

Wavelet analysis is a relatively new and promising tool for signal and image analysis. Its time-scale representation provides both spatial and frequency information, thus giving extra information compared to other image representation schemes. This research aims to address some of the problems of *query by texture* and *query by low-quality images* by exploiting all the advantages that wavelet analysis has to offer, particularly in the context of museum image collections.

A novel *query by low-quality images* algorithm is presented as a solution to the problem of poor retrieval performance using conventional methods. In the *query by texture* problem, this thesis provides a comprehensive evaluation on wavelet-based texture method as well as comparison with other techniques. A novel automatic texture segmentation algorithm and an improved block oriented decomposition is proposed for use in *query by texture*. Finally all the proposed techniques are integrated in a content-based image retrieval application for museum image collections.

Contents

List of Figures

List of Tables

Acknowledgements

The author is grateful to the Faculty of Engineering and Applied Science for the support of research studentship to undertake this work.

Also, I would like to extend my thanks to all the help my supervisor, Dr. Paul H. Lewis has given me throughout the degree, and also to my wife and my parents for being so supportive.

To my loving wife

Chapter 1

Introduction

In this first chapter, the motivation of the thesis is presented, as well as the research objectives and achievements. The outline of the entire thesis concludes the chapter.

1.1 Motivation

Recent years have seen a rapid increase in the size of digital image collections. Everyday imaging equipment generates giga-bytes of images. Surveys (1) have estimated that world-wide 2,600 new images are created per second (equivalent to 80 billion per year) with an estimated 10 billion of which are available on the internet. Finding the correct image has become an expensive problem. It is necessary to have these data organized so as to allow efficient browsing, searching, and retrieval.

Image retrieval has been a very active research area since the 1970s, with the thrust from two major research communities; database management and computer vision (2). Image retrieval can be divided into text-based image retrieval (TBIR) and content-based image retrieval (CBIR). The text-based image retrieval technique first annotates the images by text, and then uses text-based database management systems to perform image retrieval. However, there exist two major difficulties, especially when the size of image collections is large. One is the vast amount of labour required in manual image annotation. The other difficulty is the subjectivity of human perception, that is, for the same image content different people may perceive it differently. The perception subjectivity and annotation impreciseness may cause unrecoverable mismatches in later retrieval processes.

This leads to more efforts being required on content-based image retrieval. Using this technique, images are indexed by their own visual content, such as colour, texture or shape. CBIR has become more and more important with the advance of computer technology, and provides the answer to the two drawbacks of the TBIR system mentioned

above. It is important to stress that CBIR is not a replacement of, but rather a complementary component to TBIR. Only the integration of the two can result in satisfactory retrieval performance at present.

In this thesis, the application of content-based image retrieval system is applied to various museum image collections. The wavelet transform, a powerful tool in image processing and analysis will be used as the main tool for the research work.

1.2 Research Objective

The main objective of this thesis is to produce an efficient content-based image retrieval system for use with museum image collections. The museum collections are available from various museums in a collaboration with the Department of Electronics and Computer Science at the University of Southampton. Two specific areas concerned are the *query by low quality image* and *query by texture*. In *query by low quality image*, the content-based retrieval system is expected to retrieve images that are similar in content to the poor input image, for example an image obtained from a fax machine. The system should be of high accuracy while keeping the computational load at a minimum. For this reason, the wavelet transform was chosen as one of the tools to address *query by low quality image*.

In *query by texture*, the main goal of the retrieval system is to be able to retrieve images containing similar texture to the given query texture. Several texture feature extraction techniques are tested, and will also be improved in order to produce a method that is suitable for use with the mentioned image domain. After reviewing the field, wavelet-based methods are again chosen for this project because of their high accuracy and low computational load, and several wavelet-based techniques will be tested and developed to produce the best available option.

Since museum collections contain many natural scene images, texture usually appears only in some part of the image. The next objective is to find the best available approach in utilizing the texture feature extraction method to obtain local statistics. Museum images usually consists of many regions of colour and texture within the same image. Applying the feature extractor on the image globally will then result in an incorrect representation for the textures within the image. Two approaches are considered in this thesis, block oriented decomposition and automatic texture segmentation.

- *Block oriented decomposition.* Using this approach, the feature extractor is used to extract features from several sub-image blocks. Assuming the sub-image blocks are small enough and efficiently structured, this method will produce feature vectors for each and every texture present in the image. Several block oriented decomposi-

tion are studied and an improvement over the available decomposition techniques are suggested for a better localization.

- *Automatic texture segmentation.* Using this approach the image is segmented first, and the feature vectors are computed only on the textured regions. By automatic segmentation, we mean a segmenter which does not need a priori knowledge either on the type of the texture or the number of textures present in the image. A novel automatic texture segmenter must first be developed, since there are very few automatic texture segmentation algorithms available in the literature, and they are not very suitable for our application.

By the end of this stage, we will have two different approaches for content-based image retrieval using texture features. A comprehensive evaluation of both approaches is presented. It is interesting to have a comprehensive study on the advantages and the disadvantages of both approaches, hence their performance will be compared in every aspect. Because of the limited number of automatic texture segmentation algorithms available, most systems use the block oriented decomposition for extracting local texture. Therefore it would also be interesting to see if segmentation is really needed in content-based image retrieval applications.

From another point of view, although one of the main objectives of this thesis is to use texture to bring novelty to the field of content-based image retrieval, it can also bring novelty to other fields. The novel automatic texture segmentation algorithm proposed in this thesis might also be useful in remote sensing, medical image analysis and other fields associated with texture analysis.

1.3 Achievements

The original contributions in this thesis are applicable to content-based image retrieval and texture analysis applications, and can be itemized as follows:

- A novel algorithm for content-based image retrieval using low quality image as query (*query by low-quality image*),
- Evaluation of the *query by low-quality image* performance on various low-quality images,
- Performance comparison between various wavelet-based texture features, as well as other techniques,
- Identifying the best parameters to be associated with the selected wavelet-based texture feature,

- An improved texture feature extraction of the wavelet-based methods for achieving better retrieval performance,
- A survey of several block oriented decomposition techniques for use in content-based image retrieval applications,
- A modified block oriented decomposition algorithm for better localization, the multiscale algorithm,
- Comprehensive evaluation of the multiscale-based retrieval performance,
- A survey of several texture segmentation algorithms for use in content-based image retrieval applications,
- A novel automatic texture segmentation algorithm that does not need a priori knowledge on either the number or the type of textures,
- Extension of the automatic texture segmenter to produce a texture identifier, and evaluation of its integration with a content-based image retrieval system,
- Evaluation on the performance of two different *query by texture* approaches,
- Application of all the proposed algorithms on different museum image collections.

1.4 Thesis Summary

In chapter 2, a literature study of the background of the thesis area will be studied. The literature study includes the background of content-based image retrieval, texture analysis and low-quality image analysis. Since the wavelet transform will be used significantly in the thesis, the details of the wavelet transform will also be reviewed.

In chapter 3, previous work on *query by low-quality image* will be presented and discussed. A novel *query by low-quality image* algorithm is then proposed for use in museum image collections.

In chapter 4, a comprehensive comparison between several wavelet-based features as well as other texture features is conducted, and the best features are identified. The performance of the texture methods will be evaluated in terms of accuracy and computation time. Another experiment which identifies the best parameters to be associated with the best method follows. Finally an improved version of the selected texture method is proposed by examining several statistical functions to be used with the corresponding texture technique.

An improved block-oriented decomposition method is proposed in chapter 5, and its performance in achieving better localization is computed. The method is then combined

with the improved texture features method chosen in chapter 4, and its retrieval accuracy in retrieving local textures within an image is evaluated using an appropriate test set.

Chapter 6 describes several non-automatic texture segmentation algorithms available in the literature, and based on these, a novel automatic texture segmentation algorithm is proposed. The algorithm is tested on several composite textures in order to evaluate its accuracy, before evaluation on real museum images is performed. Finally an enhancement to the automatic texture segmentation is suggested to produce a texture identifier, and is integrated into a content-based image retrieval system.

Chapter 7 introduces different museum image collections available on our server, and the performances of both *query by low-quality image* and *query by texture* are evaluated on these museum databases. Discussions of the advantages and disadvantages of the proposed methods are presented. A brief description on integrating the *query by low-quality image* and the *query by texture* techniques into the Artiste and Sculpteur project is also presented.

Finally in chapter 8, the conclusions of the research are gathered together and various avenues for future works are presented.

Chapter 2

Literature Review

This chapter reviews all necessary background to gain an understanding required for this thesis. Subjects reviewed include the content-based image retrieval, the texture and low quality image analysis as well as wavelets and the wavelet transform.

2.1 Content-Based Image Retrieval

Content-based image retrieval system design highly depends on the image domain in use. In the range of images under consideration, there is a gradual distinction between narrow and broad image domains (3). In a narrow domain, one finds a limited variability of the content of the images. Usually, the recording circumstances are also very similar over the whole domain. An example of a narrow domain is a set of frontal views of faces recorded against a clear background. Although each face is unique and has large variability in the visual details, there are obvious geometrical, physical, and color-related constraints governing the domain. Another good example of a narrow domain would be a collection of fingerprint images.

In broad domains, images are polysemic and their semantics are described only partially. It might be the case that there are conspicuous objects in the scene for which the object class is unknown or even that the interpretation of the scene is not unique. The broadest class available to date is the set of all images available on the internet. Many problems of practical interest have an image domain in between these extreme ends of the spectrum. The notions of broad and narrow domains are helpful in characterizing patterns of use, in selecting features, and in designing systems. For this thesis, the image collection used is museum images, which can be classified as a medium to broad domain.

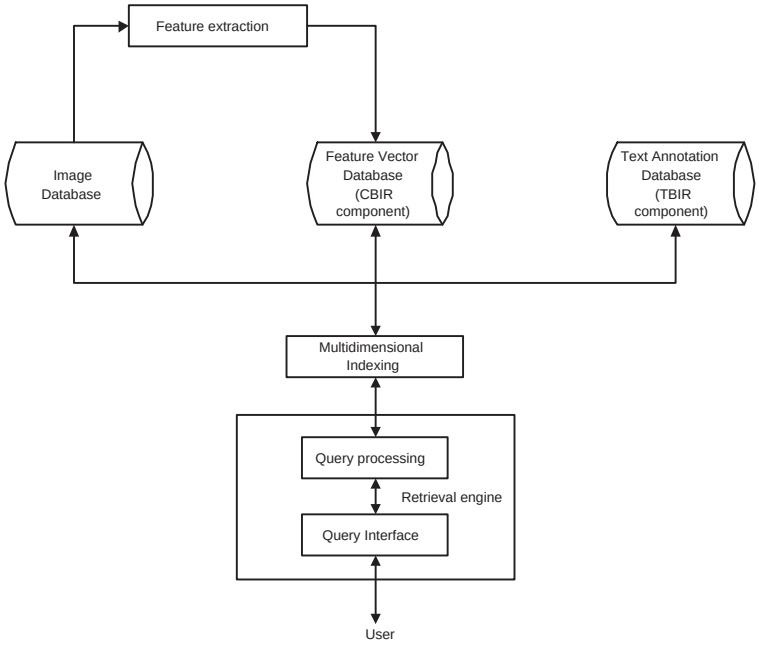


FIGURE 2.1: Image retrieval architecture.

2.1.1 CBIR System Architecture

Figure 2.1 shows a simple image retrieval system suitable for the broad image domain. From the figure, there are three databases in the system architecture, the image database, the feature vector database, and the text annotation database. The image database contains the raw images for visual display purposes. The feature vector database stores the visual features extracted from the images. This is the information needed to support CBIR. Finally the text annotation database contains the keywords and free-text descriptions of the images. The text-based retrieval is included in the system because, as mentioned earlier, only the integration of the two methods can result in satisfactory retrieval performance. However the concern of this thesis is only towards the left-hand side of Figure 2.1, i.e. the CBIR, since content-based image retrieval is still new and offers more room for improvement. In general there are two important phases in a CBIR system, which are feature extraction and feature indexing.

2.1.1.1 Feature Extraction

Feature extraction is the basis of content-based image retrieval, and features can be classified as general or domain-specific. General features are suitable for most applications and include colour, texture, shape, colour layout and shape layout features. Domain-specific features are only suitable for a narrow image domain, which is application dependent, and therefore is not of interest in the context of this thesis. Colour, texture and shape are the most used features in content-based image retrieval.

The colour feature is one of the most widely used visual features in content-based image retrieval. It is relatively robust to background complication and independent of image size and orientation. Some representative studies of colour perception and colour spaces can be found in (4; 5; 6). The color histogram is the most commonly used color feature representation. Statistically it denotes the joint probability of the intensities of the three colour channels. Histogram intersection, an $L1$ metric (metric based on absolute values), is usually used as the similarity measure for the colour histogram (7). To take into account the similarities between similar but not identical colour, Ioka (8) and Niblack *et al.* (9) introduced an $L2$-related metric (metric based on square values) in comparing histograms. An improvement of the colour histogram method includes the cumulated colour histogram (10), proposed by Stricker and Orengo. Besides the colour histogram, other colour feature representations have been applied to content-based image retrieval, including colour moments, colour sets and the colour coherence vector. Colour moments methods (10) overcome the quantization effect in the colour histogram method. Colour sets (11) is an approximation to the colour histogram, suitable for fast searching over large scale image collections. Finally the colour coherence vector (12) differs from the colour histogram in that it manages to capture information about the distribution of the colours spatially within the image.

Texture refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single colour or intensity. It is a natural property of virtually all surfaces, including clouds, trees, bricks, hair, and fabrics. It contains important information about the structural arrangement of surfaces and their relationship to their surrounding environment. Because of its importance and usefulness in pattern recognition and computer vision, there are rich research results from the past three decades. Now it further find its way into image retrieval. More and more research achievements are being added to it. A more detailed overview on texture and its features will be discussed in the next section.

Shape features involve all the properties that capture conspicuous geometric details in the image. The shape representations can be divided into two categories, boundary-based and region-based. The former use only the outer boundary of the shape while the latter use the entire shape region (13). The most successful representatives for these two categories are Fourier descriptor and moment invariants. The main idea of a Fourier descriptor is to use the Fourier transformed boundary as the shape feature. Some early work can be found in (14). A modified Fourier descriptor which is robust to noise and invariant to geometric transformation is proposed by Rui *et al* (13). The main idea of moment invariants is to use region-based moments which are invariant to transformations, as the shape feature. In (15), Hu identified seven such moments. Based on his work, many improved versions emerged, such as the Zernike moment descriptors proposed by Teague (16). The rotational invariance nature of the Zernike moment descriptors make it very useful in overcoming the shortcomings associated with Hu's

moments. Besides the Fourier descriptor and moments, some other recent work in shape representation and matching includes the finite element method (FEM)(17), the turning function (18), and the wavelet descriptors (19). The FEM defines a stiffness matrix which describes how each point on the object is connected to the other points. The turning function method is useful in comparing both convex and concave polygons. Wavelet descriptors have desirable properties such as multiresolution representation, invariance, uniqueness, stability and spatial localization.

2.1.1.2 Multidimensional Indexing

Multidimensional indexing is needed in content-based image retrieval in order to make the system truly scalable to large size image collections. It is an important issue for multimedia systems since such systems needs to handle a vast amount of multimedia data which normally involves high dimension. Similarity retrieval on multiple attributes (as opposed to exact matching) is also an important issue for multimedia systems, as most of the queries are searching for the nearest matches rather than exact matches. Many different approaches for multidimensional indexing can be found, such as the many branches of the famous tree-based algorithms (20; 21; 22; 23; 24; 25). Clustering and neural nets, widely used in pattern recognition, are also promising indexing techniques (26; 27). A thorough multidimensional indexing survey paper is given by Gaede et al (28).

The history of multidimensional indexing techniques can be traced back to the middle of 1970s, when cell methods, quad-tree, and k-d tree were first introduced. However their performances were far from satisfactory. Pushed by then urgent demand of spatial indexing systems, Guttman proposed the R-tree indexing structure (21). Based on his work, many other variants of the R-tree were developed. Sellis *et al.* proposed R+-tree in (22), while Greene proposed her variant of R-tree (23). In 1990, Beckman and Kriegel proposed the best dynamic R-tree variant, R*-tree (24). However, even for the R*-tree, it was not scalable to dimensions higher than 20 (29).

There are two main types of multidimensional indexing method: Point Access Methods and Spatial Access Methods. The Point Access Method is only for multidimensional data points whereas the Spatial Access Methods are designed for data not only represented by points but also as multidimensional spatial regions. In general, the performance of multidimensional access methods degrades dramatically with increasing dimensionality. There are studies (30; 31) that show high dimensional attributes can be represented by using a lower dimensional vector but still preserving the spatial relationship between data objects. The lower dimensional features (such as Fourier series and multidimensional scaling approaches) are then indexed by a fast multidimensional method.

However before we utilize any multidimensional indexing technique, it is beneficial to

first perform dimension reduction. At least two approaches have appeared in the literature, i.e. Karhunen-Loeve transform (KLT) and column-wise clustering. KLT and its variations, eigen-image and principal component analysis (PCA), have been studied by researchers in performing dimension reduction (20; 32; 33; 34). Experimental results form these research showed that most real data sets (visual feature vectors) can be considerably reduced in dimension without significant degradation in retrieval quality. Clustering is another powerful tool in performing dimension reduction. Row-wise clustering has been used in various disciples such as pattern recognition and speech analysis. However clustering can also be used column-wise to reduce the dimensionality of the feature space (35). Experiments show that this is a simple and effective approach.

2.1.2 Current CBIR System

This section describes some of the products which are available on the market or from research laboratories to achieve content-based retrieval on images, and other multimedia. There are far too many content-based techniques and systems to be able to describe them all here so we point the reader to reviews of the field that can be found in (36; 37; 2; 38). What follows is a brief description of some well known and well regarded systems in the field.

QBIC

QBIC (Query By Image Content) (9; 39; 40) is the first commercial and probably the most well known content-based retrieval system for image and video. Developed by IBM, its system framework and techniques have had profound effects on later image retrieval systems. The image feature extraction engine uses colour, shape and texture. The colour feature used in QBIC are the average (R,G,B), (Y,I,Q), (L,a,b), and MTM (mathematical transform to Munsell) coordinates, and a k-element colour histogram. The use of colour in QBIC was originally limited to the overall colour histogram of the image, or percentages of colour within an image, however, more recent versions of QBIC have included a widget which allows queries based on the spatial colour layout of images to be created (query by sketch). Its texture feature is an improved version of the Tamura texture representation;, i.e. combinations of coarseness, contrast, and directionality (41). Its shape feature consists of shape area, circularity, eccentricity, major axis orientation, and a set of algebraic moment invariants. QBIC is one of the few systems which takes into account the high dimensional feature indexing.

Virage

Virage (42; 43) is a system produced by Virage Inc. that performs content-based retrieval on video and images, using colour, texture, composition (colour layout), and structure (shape layout). Virage also goes one step further than QBIC, as it allows for combinations of the above to be used in a single query. Weights can be varied for each feature type according to the user's needs. The framework was also extended to include domain specific features as well as the general features.

RetrievalWare

RetrievalWare (44; 45) is a content-based image retrieval engine developed by Excalibur Technologies Corp. that, among other things, uses colour, shape, texture, brightness, colour layout, and image aspect ratio in a query-by-example paradigm to match images. Like Virage, it also supports the combinations of these features and allows the users to adjust the weights associated with each feature.

Photobook

Photobook (17) is a set of interactive tools for browsing and searching images developed at the MIT Media Lab. Photobook consists of three "subbooks" from which shape, texture, and face features are extracted, respectively. Users can then query, based on the corresponding features in each of the three "subbooks". Later versions allowed human authors to help annotate images, based on "society of models" approach that reflect the particular domain and set of users.

VisualSEEk and WebSEEk

VisualSEEk (46) is a visual feature search engine and WebSEEk is a World Wide Web oriented text/image search engine, both of which are developed at Columbia University. Main research features are spatial relationship query of image regions and visual feature extraction from the compressed domain. The visual features used in their systems are colour set and wavelet transform based texture features. To speed up the retrieval process, they also developed binary tree based indexing algorithms. VisualSEEk supports queries based on both visual features and their spatial relationship. WebSEEk is a web oriented search engine. It consists of three main modules, i.e. image.video collecting module, subject classification and indexing module, and search, browse and retrieval module. It supports queries based on both keywords and visual content.

Netra

Netra (47) is a prototype image retrieval system developed in the UCSB Alexandria Digital Library project. Netra uses colour, texture, shape, and spatial location information in the segmented image regions to search and retrieve similar regions from the database. main research features of the Netra system are its Gabor filter based texture analysis, neural net-based image thesaurus construction, and edge flow-based region segmentation.

MARS

MARS (multimedia analysis and retrieval system) (48), developed at University of Illinois at Urbana-Champaign, is a multimedia retrieval system that is designed to retrieve text, images and video.The image retrieval engine they use is based on colour and texture. Colour histogram intersection and colour moments are used to match whole images, as well as co-occurrence matrix and wavelet based methods of texture analysis. MARS also demonstrates some unique features such as the integration of database management system and information retrieval, the integration of indexing and retrieval, and integration of computer and human.

eVe

eVe (the eVision Visual Engine) (1), is a commercial product developed by eVision, LLC technologies. The engine uses automatic segmentation techniques applied to colour, texture, shape, and visual and text meta-tag searching. It attempts automatic segmentation by grouping pixels based on pixel similarity and labels the clusters as objects. Although they profess that this "brings unsupervised segmentation to the commercial world", many of their examples contain objects on white background, and those which do not are poorly segmented.

PicToSeek

PicToSeek (49) is a content-based image search system, designed for use on the web by the Intelligent Sensory Information Systems research group, at the University of Amsterdam. The system uses a colour model that is colour constant - that is, it is independent of the illumination colour, shadows, and shading cues. PicToSeek, however, is only concerned with the whole image histograms, and does not allow spatially oriented queries.

Color-WISE

Color-WISE (50) is an image similarity retrieval system which allows users to search for stored images using matching based on the localized dominant hue and saturation values. It uses a cunning fixed segmentation of overlapping elements to ensure that the matching is slightly fuzzy. The system computes separate histograms for hue, saturation and intensity, and reduces their size by finding their area-peak - basically removing noise that is small amount of isolated colours. Color-WISE uses Microsoft Access to perform the database functions, and uses a similarity metric based on QBIC system. Querying in Color-WISE is achieved with query-by-image.

Blobworld

Blobworld (51) was developed at the University of Berkeley, California under the Digital Library Project. It uses low-level grouping techniques to create "blobs of stuff", which can be texture, colour or symmetry. The blobs can be matched against their content, and their position, and it is possible to use high-level techniques to analyse the semantics of the blobs (such as where they are in relation to other blobs), and conclude what they might represent.

Image-MINER

Image-MINER (52) is an image and video retrieval system developed by the AI group at the University of Bremen. Their colour indexing system for images uses local histograms in a fixed grid geometry. Further grouping of the fixed elements occurs to get 'color-rectangles', which are signatures for their input images. The colour based segmentation module, is part of the larger Image-MINER system which includes video retrieval methods, including shot detection and subsequent 'mosaicing'.

Other Systems

ART MUSEUM (53), developed in 1992, is one of the earliest content-based image retrieval systems. It uses the edge feature as the visual feature for retrieval. CAETIIML (54), built at Princeton University, uses a combination of the on-line similarity searching and off-line subject searching. Some other systems include IMatch (55) by mwlabs which uses colour, texture and shape for image retrieval, and DART (56) by AT&T which also uses colour, texture, shape as well as locally smooth regions.

Summary

There are many content-based retrieval systems which have been developed within the last decade using relatively simple matching techniques. As described above, most of these systems use colour, texture, shape and some have basic spatial location retrieval (low-level features). However, few of them access the semantics (high-level features) of the image and therefore fail to allow retrieval within the semantic domain. Very recently in the literature, semantic retrieval is becoming an active research topic (57; 58; 59). Should the above retrieval system be armed to support semantic-based querying, it is highly likely that the retrieval performance will be much improved. However, having said that, the role of low level features in image retrieval should not be overlooked. Instead there must be a proper research balance between the two medium in order to make content-based retrieval successful.

2.2 Low-Quality Image Analysis

It is interesting to note that, in published works on image retrieval, almost all efforts are spent on normal quality image retrieval, that is using images captured digitally from camera or video, or at least images obtained from a scanner as query. There are actually other image acquisition media which can provide query images, but with a rather poor quality; a fax machine is a very good example. This section briefly explains the idea of *query by low-quality images*, and why it is necessary.

The motivation for research on low-quality image comes from a requirement by some museums to respond to queries for pictorial information, submitted in the form of fax messages or other low-quality monochrome images of works of art. The museums have databases of high-resolution images of their artefact collections and the person submitting the query is asking typically whether the museum holds the art work shown or perhaps some similar work.

Typically the query image will have no associated metadata and will be produced from a low-resolution picture of the original art work. The resulting poor quality image, received by the museum, leads to very poor retrieval accuracy when the fax is used in standard *query by example* searches using, for example, colour or texture matching algorithms. Some examples of genuine fax images received by the museum together with the corresponding high-resolution images they represent are shown in Figure 2.2. It is obvious that fax images are of very low quality and even the best quality of fax machine cannot prevent the loss of information in the fax image. If this image is to be used as query, special algorithms need to be developed in order to improve the retrieval accuracy.

With the advance of imaging and communication technology these days, one might

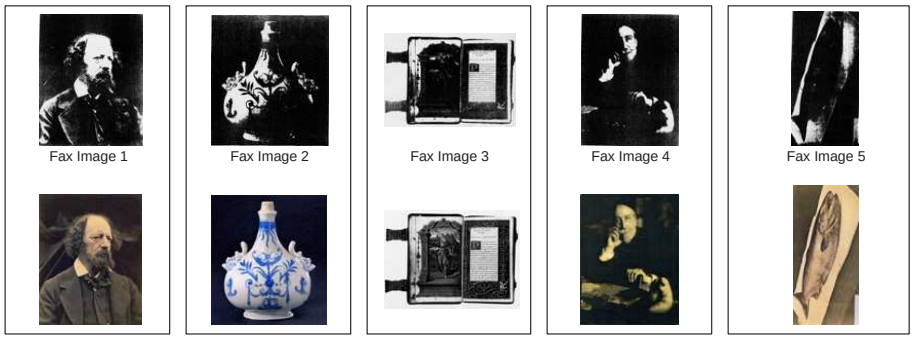


FIGURE 2.2: Examples of fax images and their originals

wonder if anyone would still use fax machines as a medium to transfer images. The answer is yes; there is a study reporting that even with the popularity of e-mail and photo-messaging, faxing is more popular than it has ever been (60). In the context of this thesis, there are a lot of museums which hold large databases of images and keep receiving fax images from all around the world asking if they have the artwork shown. They have problems in fulfilling requests as a result of poor retrieval results using standard *query by example* searches. Moreover even images captured by camera or scanner sometimes carry noise, although they are not as bad as in fax images. Therefore this specific area of content-based image retrieval needs to be explored more comprehensively.

2.3 Texture Analysis

In many machine vision and image processing algorithms, simplifying assumptions are made about the uniformity of intensities in local image regions. However, images of real objects often do not only exhibit regions of uniform intensities. For example, the image of a wooden surface is not uniform but contains variations of intensities which form certain repeated patterns called visual texture. The patterns can be the result of physical surface properties such as roughness or oriented strands which often have tactile quality, or they could be the result of reflectance differences such as colour on a surface.

Texture analysis is an important and useful area of study in machine vision. Most natural surfaces exhibit texture and a successful vision system must be able to deal with the texture world surrounding it. Texture analysis methods have been utilized in a variety of application domains. In some of the mature domain (such as remote sensing) texture already has played a major role, while in other disciplines (such as surface inspection) new applications of texture are being found. Some examples of the fields that texture plays a major role are:

- Remote Sensing. Texture is extensively used in land use classification where ho-

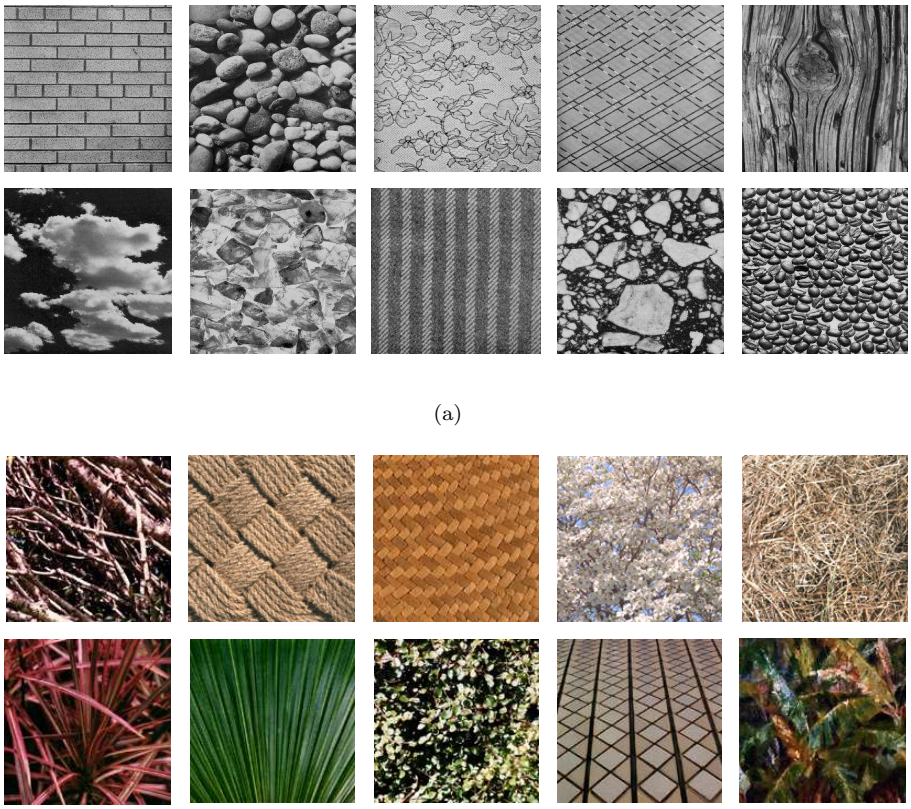
mogeneous regions with different types of terrains (such as wheat, bodies of water, urban regions, etc.) need to be identified.

- Medical Image Analysis. The applications involve the automatic extraction of features from the image which are then used for a variety of classification tasks, such as distinguishing normal tissue from abnormal tissue.
- Surface Inspection. The applications include defect detection in image of textiles and automated inspection of carpet wear and automobile paints.
- Document Processing. Texture is used as a tool to segment document images to identify regions of interest.
- Image and Video Retrieval. Texture is one of the main features used for measuring similarity between multimedia data.

2.3.1 Definition of Texture

We recognize texture when we see it but it is very difficult to define. This difficulty is demonstrated by the number of different texture definitions attempted by vision researchers. Coggins (61) has compiled a catalogue of texture definitions in the computer vision literature and some examples are given below:

- "We may regard texture as what constitutes a macroscopic region. Its structure is simply attributed to the repetitive patterns in which elements or primitives are arranged according to a placement rule." -Tamura *et al.* (41)
- "A region in an image has a constant texture if a set of local statistics or other local properties of the picture function are constant, slowly varying, or approximately periodic." - Sklansky (62)
- "The image texture we consider is nonfigurative and cellular. An image texture is described by the number and types of its (tonal) primitives and the spatial organization or layout of its (tonal) primitives. A fundamental characteristic of texture: it cannot be analyzed without a frame of reference of tonal primitive being stated or implied. For any smooth gray-tone surface, there exists a scale such that when the surface is examined, it has no texture. Then as resolution increases, it takes on a fine texture and then a coarse texture." - Haralick (63)
- "The notion of texture appears to depend upon three ingredients: (i) some local 'order' is repeated over a region which is large in comparison to the order's size, (ii) the order consists in the non-random arrangement of elementary parts, and (iii) the parts are roughly uniform entities having approximately the same dimensions everywhere within the textured region." -Hawkins (64)



(a)

(b)

FIGURE 2.3: Examples of texture (a) Brodatz texture , (b) Vision texture

This collection of definitions demonstrates that the "definition" of texture is formulated by different people depending upon the particular application and that there is no generally agreed upon definition. Some are perceptually motivated, and others are driven completely by the application in which the definition will be used.

Some examples of texture are given in Figure 2.3 where the textures are taken from the Brodatz collection and the Vision texture collection. The Brodatz texture collection consists of 112 textures with wide varieties and is widely used as a standard texture analysis platform. Vision texture (VisTex) collection was made as an alternative to the Brodatz texture library, which is not freely available for research use. However, unlike the Brodatz collection, the images in VisTex do not conform to rigid frontal plane perspectives and studio lighting conditions

2.3.2 Texture Properties

In spite of the lack of a general definition of texture in the computer vision literature, there are a number of intuitive properties of texture which are generally assumed to be true (65).

- Texture is a property of areas; the texture of a point is undefined. So texture is a contextual property and its definition must involve gray values in a spatial neighbourhood. The size of this neighbourhood depends upon the texture type, or the size of the primitives defining the texture.
- Texture involves the spatial distribution of gray levels. Thus, two-dimensional histograms or co-occurrence matrices are reasonable texture analysis tools.
- Texture in an image can be perceived at different scales or levels of resolution. For example, consider the texture represented in a brick wall. At a coarse resolution, the texture is perceived as formed by the individual bricks in the wall; the interior details in the brick are lost. At a higher resolution, when only a few bricks are in the field of view, the perceived texture shows the details in the brick.
- A region is perceived to have texture when the number of primitive objects in the region is large. If only a few primitives objects are present, then a group of countable objects is perceived instead of a textured image. In other words, a texture is perceived when significant individual "forms" are not present.

2.3.3 Texture Feature Extraction Method

A texture feature is a value, computed from the image of an object, that quantifies some characteristic of the grey level variation within the object. There are many feature extraction techniques developed in order to improve the classification of textured images. However, due to the fact that the perception of texture has so many different dimension, there is no single method of texture representation which is adequate for a variety of textures (65). Texture feature extraction methods can be divided into statistical, model-based, signal processing and geometrical categories. However the use of geometrical (structural) methods are rather limited because of tight assumptions of the nature of textures. The methods briefly described below are all categorized into either statistical, model-based or signal processing methods as they are the most widely used.

In the early 1970s, Haralick et al. (66) proposed the co-occurrence matrix representation of texture features. This approach explored the gray level spatial dependence of texture. Meaningful statistics are extracted from the matrix as the texture representation. Many other researchers followed the same line and further proposed enhanced versions. Motivated by the psychological studies in human visual perception of texture, Tamura et al. (41) explored the texture representation from a different angle. They developed computational approximations to the visual texture properties found to be important in psychology studies. The six visual texture properties were coarseness, contrast, directionality, line-likeness, regularity, and roughness. The QBIC system (39) and the MARS system (48) further improved this texture representation.

The Markov Random Field model (MRF) (67) assumes that the state of an image pixel is highly dependent on the brightness value and the configuration of neighbouring pixels, and this assumption is exploited to get texture information. The Gaussian-Markov Random Field (GMRF) and the multiresolution GMRF are the modified and improved versions of MRF. In (68), Chellappa and Chatterjee use the GMRF model to classify GMRF-generated textures. Another method that received intensive research is the simultaneous auto-regressive model (SAR) (69; 70). Like the Markov Random field method and its families, the SAR method used the information from the neighbouring pixels to compute the brightness of a particular pixel, but with error prediction properties.

In (71), Pentland suggested the use of fractal dimension for texture analysis. Briefly, 'fractal analysis for textures' refers to the search for the number that best characterizes the assumed self-similar distribution of gray scale values, which comprise the texture (72). An improved fractal-based analysis is proposed by Chauduri and Sarkar (73) using the multi-fractal concept, in which, instead of using only one FD per pixel block, they used 6 FDs from 6 modified original images. The Gabor transform (74; 75) is another popular method for texture analysis. The basic idea behind the Gabor method is to compute the features based on scale and orientation, where different textures have energy concentrated in different scales and orientations.

In the early 1990s, after the wavelet transform was introduced and its theoretical framework was established, many researchers began to study the use of the wavelet transform as texture features. In (76), Smith and Chang used the mean and variance extracted from the wavelet channels as texture features. To explore the middle band characteristics, a tree-structured wavelet transform was used by Chang and Kuo (77) to improve the classification accuracy. To achieve translational invariance, Unser (78) proposed a discrete wavelet frames as the solution. The wavelet transform was also combined with other techniques to achieve better performance, such as with the co-occurrence matrix (79; 80).

Other methods that have been used for texture feature extraction include the logical operators method (81), Jordan features (82), the Statistical Geometrical Features (SGF) (83), the Laws texture feature (84), the kernel Principal Component Analysis (85), the Geodesic active contours (86), the Voronoi polygons (87), and the edge-based method (88). There are also quite a few review papers in this area. An early review paper by Weszka et al. (89), compared the texture classification performance of Fourier power spectrum, co-occurrence matrix, and first-order statistics of grey level differences. They found out that the Fourier method performs poorly, while the other two were comparable. In (90), Ohanian and Dubes compared and evaluated four types of textural features, namely Markov random field, multichannel filtering, fractal-based and co-occurrence matrix. They found that the co-occurrence matrix performed best in their test set. Ma and Manjunath (91) evaluated various wavelet-based features, and found that the Gabor

wavelet was the best among the tested candidates.

2.4 Wavelets and The Wavelet Transform

For many years the Fourier Transform has been the most popular time-frequency representation of a signal. A Fourier transform expands a signal using a set of periodical functions. The resulting coefficients are usually presented as a combination of two signals. The magnitude signal contains the frequency content, since the functions represent different frequencies. The phase signal contains the spatial information in an implicit way which cannot be interpreted directly.

Because of this, Fourier analysis has a serious drawback. In transforming to the frequency domain, spatial information is lost. When looking at the Fourier transform of a function, it is impossible to tell when a particular event took place. Although the Windowed Fourier Transform (or Short-Time Fourier Transform) manages to represent a sort of compromise between the time- and frequency-based views of a signal, they introduce yet another problem, that is the reduction in flexibility caused by the fixed window size.

Wavelet analysis, however, does not suffer from these problems. A wavelet transform expands a function using a set of wavelet functions. The resulting wavelet coefficients contain the frequency content since the wavelet functions also represent different scales (or frequencies). The wavelet functions are also localised spatially, which means that the expansion is local: coefficients are computed for all points of the original signal, thus the spatial information remains intact.

2.4.1 Multiresolution and Wavelets

Multiresolution techniques for signal and image processing have been in use for decades (92). The need for transforms that lead to a representation in which both spatial and frequency information are present, is widely recognized. In order to provide this, a lot of related techniques were developed, including Gabor, Walsh-Hadamard and subband filtering, to name a few. Around 1990, a mathematical framework emerged which provides a more formal, solid and unified approach to multiresolution representation (93).

The wavelet representation encompasses multiresolution decompositions of signals into orthogonal bases of wavelet functions which have compact support (exactly zero outside an interval). The development of methods to construct such functions is an important aspect of the theory. In fact, the advent of smooth and compact wavelets initiated the breakthrough of the framework, since they made it possible to compute sufficiently precise decompositions with limited computational effort (94). In their brief history

within the signal and image processing field, wavelets have already proven themselves to be an indispensable addition to the analyst's collection of tools and continue to produce successful research work. It is now a preferred tool which provides both conceptual and computational advantages compared to other techniques.

2.4.2 Definition of Wavelets

Wavelets can be described in more than one way. From the mathematical point of view, wavelets are a set of basis functions. This set of functions is generated from the dilations and translations of a unique function called the mother wavelet, ψ. Any function can be expressed in terms of wavelets. This means that given a mother wavelet, we can project any function onto the dilated and translated versions of this mother wavelet, much like the sine and cosine functions in the Fourier domain.

From the engineering point of view, wavelets are band pass filters. The wavelet basis can be viewed as a bank of filters with various bandwidths. The complete set of the wavelet filter bank covers all frequencies in the Fourier domain and therefore all frequency elements of a signal can be extracted from a set of wavelets. From the physicist's point of view, the wavelet functions are a set of functions which have good localisation in both the time and frequency domains, which means that the wavelet transform can perceive both frequency and time information from a given signal.

2.4.3 Properties of Wavelets

Having been exposed to the idea of wavelets, it is important to know the essential characteristics of wavelets. A wavelet is a function ψ (95) whose Fourier spectrum $\hat{\psi}$ satisfies the admissibility criterion:

$$\int_{-\infty}^{+\infty} \frac{|\psi(\hat{t})^2|}{|t|} dt < +\infty \tag{2.1}$$

If $\psi(t)$ is sufficiently regular (this means that it decreases exponentially for increasing t), this condition reduced to:

$$\hat{\psi(0)} \Longleftrightarrow \int_{-\infty}^{+\infty} \psi(t) dt = 0; \tag{2.2}$$

which means that a wavelet has to be localised (must oscillate and decay) and have zero mean (must integrate to zero). A large variety of wavelet functions can be constructed. Examples of some well known wavelets are shown in Figure 2.4. The main design choice that has to be made is whether the localisation should be strongest in the frequency or the spatial domain.

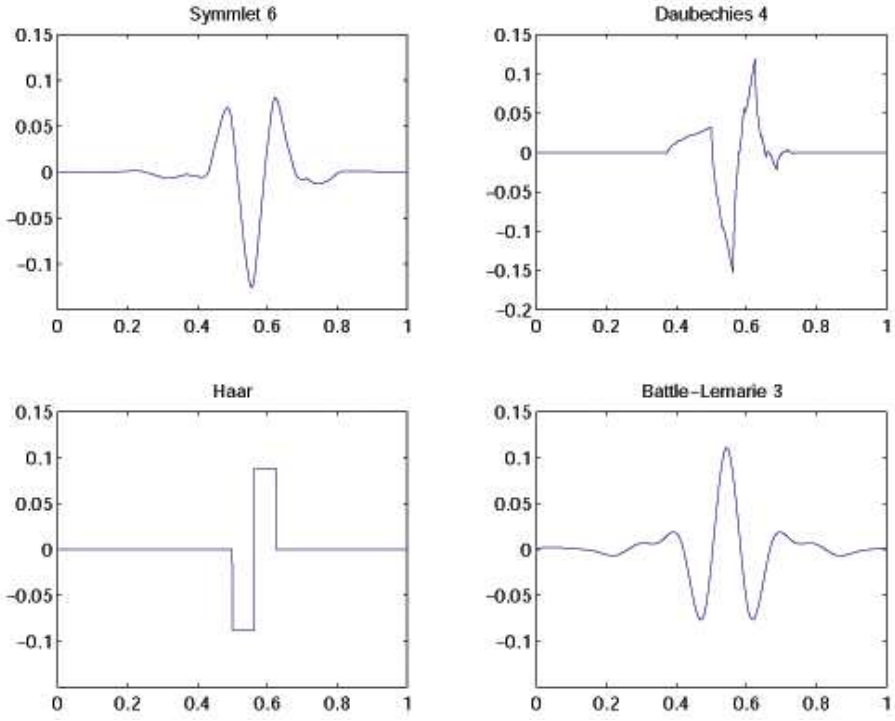


FIGURE 2.4: Example of one-dimensional wavelet families. The number after each family name corresponds to the number of vanishing moment.

Wavelets comprise an infinite set. The different wavelet families make different trade-offs between several wavelet properties such as regularity, symmetry, orthogonality and compactness (in both time and frequency). Within each family of wavelets (such as the *Daubechies* family) are wavelet subclasses distinguished by the number of coefficients and by the level of iteration. Wavelets are classified within a family most often by the number of vanishing moments. For example, within the *Coiflet* wavelet family are *Coiflet* with two vanishing moments and *Coiflet* with three vanishing moments. A wavelet has m vanishing moments if:

$$\int_{-\infty}^{\infty} t^l \psi(t)\, dt = 0 \qquad \text{for } l = 0, 1, ..., m-1. \tag{2.3}$$

The vanishing moments are directly related to the number of coefficients and the smoothness of wavelets. The larger the number of vanishing moments, the smoother the wavelets. There is also an additional function associated with some, but not all wavelets. This is the so-called scaling function or father wavelet, ϕ, and, along with the orthogonality property, determines whether a particular wavelet family can be considered for discrete wavelet transform.

Based on the above properties, wavelets can be divided into five different categories (96):

- Crude wavelets (have only minimal properties), such as the Gaussian, Morlet and

Mexican Hat wavelet families.

- Infinitely regular wavelets, such as the Meyer wavelet family.
- Orthogonal and compactly supported wavelets, such as the *Haar*, *Daubechies*, *Coiflet* and *Symlet* wavelet families.
- Bi-orthogonal and compactly supported wavelet pairs, such as B-spline wavelets.
- Complex wavelets, such as the Gabor wavelet family.

Each category holds different properties and has its own advantages and disadvantages. The most notable difference is the fact that only the third and fourth category are qualified to use the fast algorithm discrete wavelet transform, which will be discussed later.

2.4.4 One-Dimensional Wavelet Transform

As stated earlier, wavelets are dilations and translations of the mother wavelet, ψ. These dilations and translations are derived from the formula:

$$\psi_{(s,l)}(t) = s^{-1/2}\psi(st - l) \tag{2.4}$$

for any real numbers s and l, where s and l are variables that dilate and translate the mother function (95).

The wavelet transform is the process of transforming a particular signal into a time-scale representation of it. It is easier to explain wavelet transform by comparing it to the Fourier transform. Mathematically, the process of Fourier analysis is represented by the Fourier transform:

$$F(w) = \int_{-\infty}^{\infty} f(t)e^{-jwt}dt \tag{2.5}$$

which is the sum over all time of the signal $f(t)$ multiplied by a complex exponential. The results of the transform are the Fourier coefficients $F(w)$, which when multiplied by a sinusoid of frequency w, yield the constituent sinusoidal components of the original signal.

Similarly, wavelet transform is defined as the sum over all time of the signal multiplied by scaled, shifted versions of the mother wavelet function ψ:

$$C(scale, position) = \int_{-\infty}^{\infty} f(t)\psi(scale, position, t)dt \tag{2.6}$$

The results of the wavelet transform are many wavelet coefficients, C, which are a function of scale and position. Multiplying each coefficient by the appropriately scaled and shifted wavelet yields the constituent wavelets of the original signal.

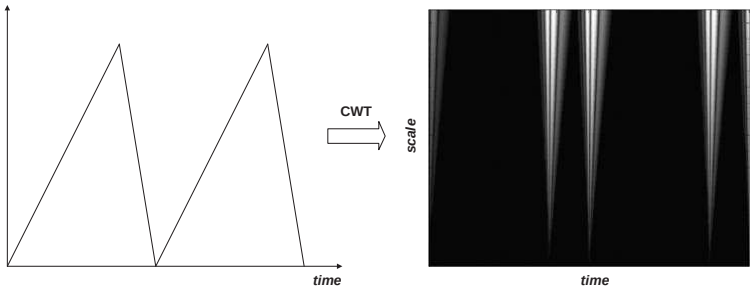


FIGURE 2.5: A signal and its wavelet transform

Note that C is a coefficient with respect to scale and position. Therefore the output of wavelet transform will have one extra dimension from the original signal. For a one-dimensional signal, the wavelet transform output will have two dimensions, while for a two-dimensional signal such as image, the wavelet transform output will have three dimensions, and can be viewed as an image stack. An example of applying wavelet transform to a one-dimensional signal is shown in Figure 2.5. It is a different view of signal data from the time-frequency Fourier view, but it is not unrelated. The higher scales correspond to the most "stretched" wavelets. The more stretched the wavelet, the longer the portion of the signal with which it is being compared, and thus the coarser the signal features being measured by the wavelet coefficients. Thus, there is a correspondence between wavelet scales and frequency as revealed by wavelet analysis:

- Low scale ⇒ Compressed wavelet ⇒ Rapidly changing details ⇒ High frequency
- High scale ⇒ Stretched wavelet ⇒ Slowly changing, coarse features ⇒ Low frequency .

It's important to understand that the fact that wavelet analysis does not produce a time-frequency view of a signal is not a weakness, but a strength of the technique. Not only is time-scale a different way to view data, it is a very natural way to view data deriving from a great number of natural phenomena. The wavelet transform coefficient plot has proved to be a much better view of data than the traditional time-frequency view.

2.4.4.1 Discrete Wavelet Transform (DWT)

Calculating wavelet coefficients at every possible scale is computationally expensive, and it generates a lot of data. What happens if we choose only a subset of scales and positions at which to make our calculations? It turns out, rather remarkably, that if we choose scales and positions based on powers of two, so-called dyadic scales and positions, then our analysis will be much more efficient and just as accurate (93). Such an analysis is referred to as discrete wavelet transform (DWT).

However not all wavelets are suitable for this kind of analysis. To ensure a lossless representation and a perfect reconstruction of data, the mother wavelet, ψ must be associated with a scaling function, ϕ, and both ψ and ϕ have to be orthogonal and compactly supported. As mentioned earlier, only some wavelets are associated with a scaling function. Unlike the mother wavelet which integrates to zero, the scaling function must integrate to unity. The dilation and translation must now be performed on both the mother wavelet and the scaling function.

$$\begin{aligned} \psi_{(s,l)}(t) &= 2^{-s/2}\psi(2^{-s}t - l) \\ \phi_{(s,l)}(t) &= 2^{-s/2}\phi(2^{-s}t - l) \end{aligned} \tag{2.7}$$

In other words, to perform DWT, there must be a pair of functions, one having a zero *dc* component and the other having a unit *dc* component, the dilations and translations of which form a complete basis for the lossless representation of a signal.

Figure 2.6 shows the dilations and translations of a wavelet in discrete wavelet transform. The Haar wavelet, which is an odd rectangular pulse pair, is the simplest and oldest orthonormal wavelet with compact support, and will be used for illustration purposes. Starting from the finer scales, the basic wavelet is translated by increments equal to its width, so that the complete set of wavelets at any scale completely covers the interval. The basic wavelet is progressively stretched (increased in scale) by powers of two. As the basic wavelet is scaled up by powers of two, its amplitude is scaled down by powers of $\sqrt{2}$, to maintain orthonormality. The result of this is a set of orthonormal basis functions. The scaling function completes the interval in the coarsest scale.

An efficient way to implement discrete wavelet transform using filters was developed in 1988 by Mallat (93). This very practical filtering algorithm, which is based on the theory of multiresolution analysis, yields a fast discrete wavelet transform, a box into which a signal passes, and out of which wavelet coefficients quickly emerge. In the paper, the author verified that a family of orthogonal wavelets and their scaling function can be represented by a bank of quadrature mirror filters (QMF), making it possible to replace ψ and ϕ with the QMF itself in computing the wavelet coefficients.

The quadrature mirror filter is a special filter that can act as both a high and a low-pass filter H and L by re-arranging its coefficients. Filtering a signal with a low-pass QMF produces the coefficients corresponding to the mother wavelet, also termed *approximation coefficients*. Similarly, filtering a signal with a high-pass QMF produces the coefficients corresponding to the scaling function, also termed *detail coefficients*. The same QMF filter is used during reconstruction, i.e. converting the coefficients back to obtain the original data. This process of high and low-pass filtering of signal to get two sets of data is called signal decomposition. The resulting high and low-pass signals can be sub-sampled by a factor of 2, without loss of information. The filtering and

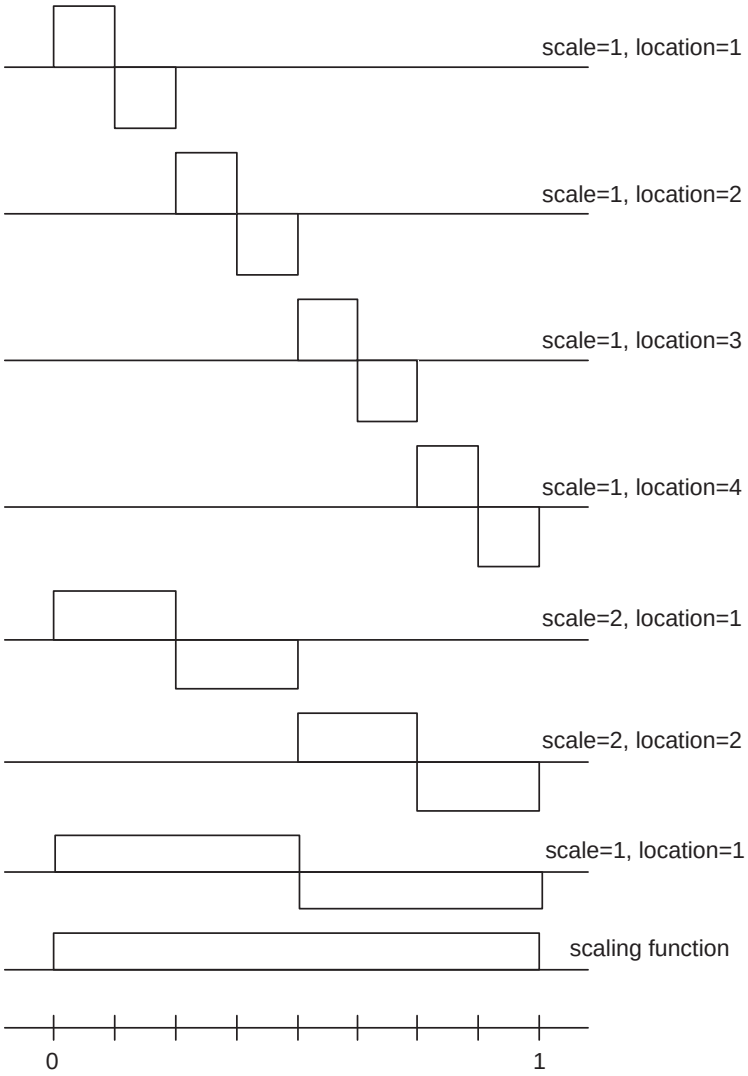


FIGURE 2.6: The Haar wavelet basis functions

sub-sampling is repeated iteratively on the low-pass signal. The result of this process is a pyramid of signals with successive frequency content, which is called standard or pyramidal wavelet transform.

Alternatively, the decomposition can be iterated not only on the low-pass but on the high pass signals as well. This results in a binary tree of signals which is called tree-structured wavelet transform or wavelet packet transform. Because of the sub sampling, the total representation contains the same number of samples as the original signal. When sub-sampling is omitted, an over-complete representation is obtained, which is called discrete wavelet frames. The filtering approach is most widely used because it is easy to understand and familiar in a signal processing context. The results of the decomposition depend on the filters (or corresponding wavelets) that are used. In practise, usually it is the filters that are constructed in such a way that their corresponding functions have the desired wavelet properties.

2.4.4.2 Continuous Wavelet Transform (CWT)

Without the discretisation step, one deals with a continuous wavelet transform (CWT) instead of a DWT. This is more general; it allows continuous scale selection and a more flexible filter design. However, there exists no efficient implementation scheme similar to that of the DWT. In fact, they are usually implemented in the same way as continuous Gabor transform, by filtering in the Fourier domain. The difference between CWT and Gabor transform lies only in the properties of the basis functions. While important in theory, this difference can become negligible in practise. For example, the B-spline wavelet has a corresponding Gabor function from which it differs only by small correction term. This is also why the Gabor transform is sometimes referred to as the Gabor wavelet transform. So in practical applications, the main design choice is to make between discrete transforms, which are fast, or continuous transforms which offer more precision and additional fine-tuning possibilities.

2.4.5 Two-Dimensional Wavelet Transform

To apply wavelet transform in images, the extension to two-dimensional has to be made. For two-dimensional data, the transform can be categorized into separable and non-separable transform. For the non-separable wavelet transform, the wavelet basis used is a nonseparable function. This results in the transform being performed by filtering in the Fourier domain, as in the continuous wavelet transform. This requires expensive computation and therefore have no advantage over other type of transform computationally. Most discrete wavelet transform in two-dimensional data is performed using the separable wavelet transform. For the separable transform, the two-dimensional wavelets are defined as tensor products of one-dimensional wavelets. This results in one scaling function and three different mother wavelets:

$$\begin{aligned} \phi[i,j] &= \phi[i]\phi[j] \\ \psi_1[i,j] &= \phi[i]\psi[j] \\ \psi_2[i,j] &= \psi[i]\phi[j] \\ \psi_3[i,j] &= \psi[i]\psi[j] \end{aligned} \tag{2.8}$$

This type of product allows one-dimensional filtering of rows, followed by one-dimensional filtering of columns, instead of two-dimensional filtering. This results in four different filtered images; one of which represent the *approximation*, namely the *low-low* (LL) channel, while the other three represent the *details*, namely the *low-high* (LH) channel, *high-low* (HL) channel and *high-high* (HH) channel. LL is a smoothed version of the original image I, while the detail images LH, HL and HH contain respectively the details of the vertical, horizontal and diagonal directions, thus retaining specific orientational information. After this, a sub-sampling in both directions can be performed.

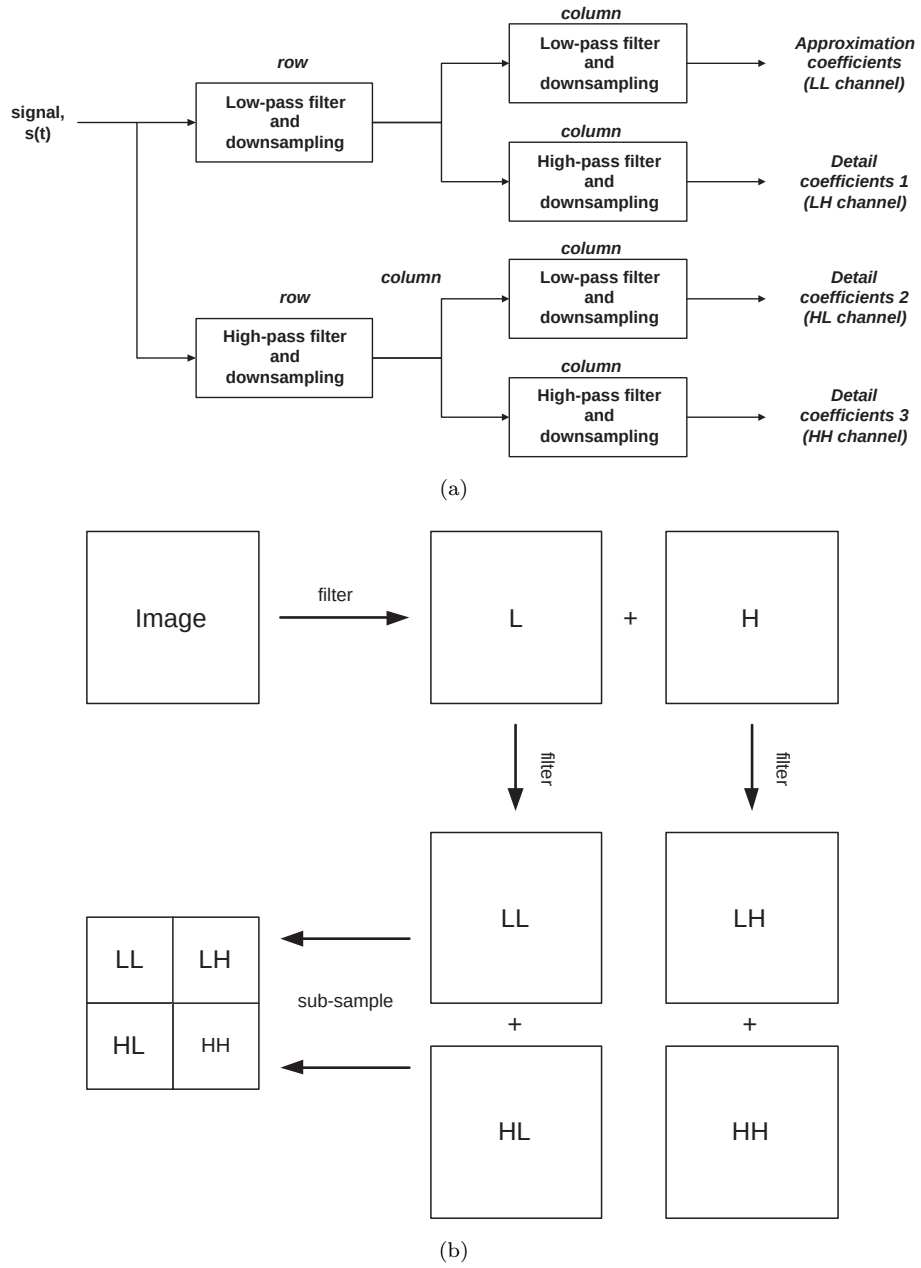


FIGURE 2.7: One level of wavelet decomposition of two-dimensional data

Figure 2.7 illustrates the filtering algorithm on two-dimensional data.

As in the one-dimensional case, by iterating the procedure on successive low-pass images LL, sub-images LL on different levels are generated. This results in a pyramid with detail images for different scales and orientations, the two-dimensional standard or pyramidal wavelet decomposition. When not only the LL, but other sub-images are decomposed further, a tree-structured wavelet transform or wavelet packet for two dimensions are obtained. If the sub-sampling is not performed, the discrete wavelet frames are obtained, and can be viewed as a stack of wavelet images. All are depicted in Figure 2.8.

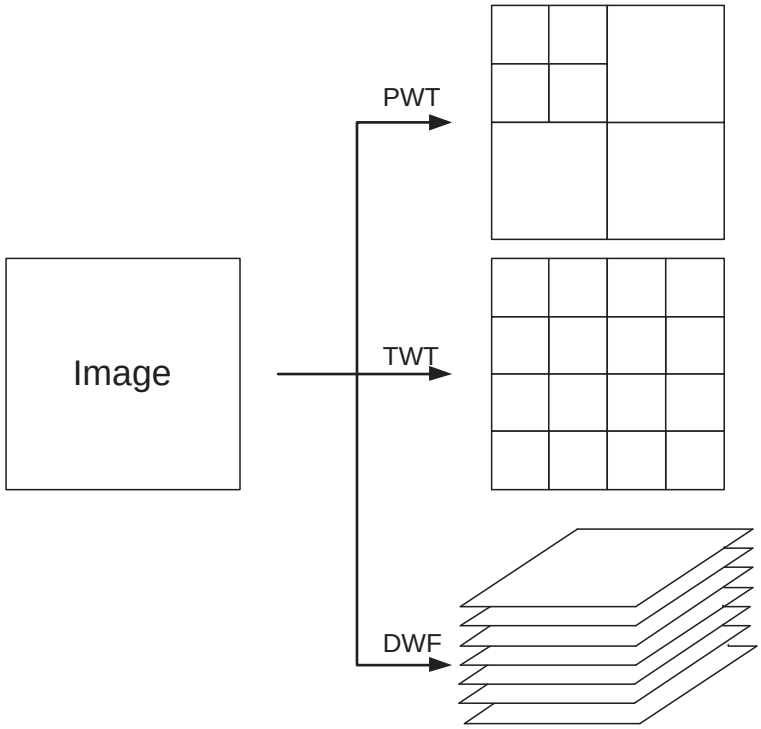


FIGURE 2.8: First two levels of a pyramidal, tree-structured and wavelet frames decomposition

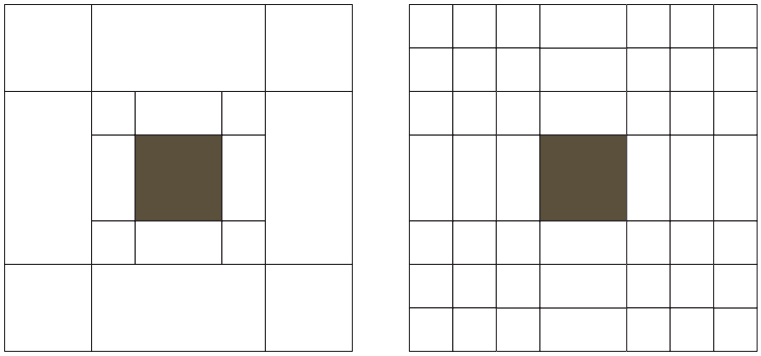


FIGURE 2.9: Frequency splitting for (left) 2-level pyramidal and, (right) tree-structured wavelet transform

The frequency splitting of the standard wavelet transform is in octave bands. This leads to a description that is sometimes too coarse, especially for higher frequencies. Using a wavelet packet transform refines the frequency splitting for the higher frequencies , at the expense of leading to a large number of sub-images and worse spatial localisation. It is instructive to look at the frequency splitting in Fourier space, as shown in Figure 2.9

2.4.6 Computational Complexity

The wavelet transform essentially involves filtering with two one-dimensional convolution masks. Therefore one level of decomposition has a computational complexity which is linear in n, the number of image pixels. For a multilevel decomposition, the depth d also determines the complexity. It is limited by the size of the image: $d \leq \frac{1}{2} log_2 n$. The computational complexity becomes:

$$O(n, d) = n \sum_{i=0}^{i<d} C \tag{2.9}$$

with C the fraction of the image pixels taken to a next level. For $C \neq 1$:

$$O(n, d) = n \frac{C^d - 1}{C - 1} = n \frac{C^{\frac{1}{2} log_2 n} - 1}{C - 1} \tag{2.10}$$

Therefore, for the three cases of decomposition described above, the computational complexity is as below:

- standard wavelet transform: $< \frac{4}{3} n$
- tree-structured wavelet transform: $n * d$
- discrete wavelet frames: $n * d$

The wavelet based methods compare favorably with Fourier based frequency analysis, since the fast Fourier transform has a complexity of n log n. Because of the advantage in computational complexity as well as better localisation in both time and frequency domain, wavelet is a very powerful tools for image processing and analysis, and therefore will be our main tools in this thesis.

Chapter 3

Low-Quality Image Analysis

This chapter starts with a review on low-quality image analysis, and the motivation behind the whole project. Since fax images are the reason for the exploration of this field, they will be used as the main source of low-quality images throughout this chapter. The properties of fax images as well as other low quality images are analysed and discussed. A novel algorithm for *query by low-quality image* application is then proposed, and its performance is evaluated on a collection of museum images using fax images as well as some other low quality image types as query.

3.1 Introduction

As mentioned in chapter 2, the motivation for research on low-quality images comes from a requirement by some museums to respond to queries for pictorial information, submitted in the form of fax messages or other low-quality monochrome images of works of art. The museums have databases of high-resolution images of their artefact collections and the person submitting the query is asking typically whether the museum holds the art work shown or perhaps some similar work. Typically the query image will have no associated metadata and will be produced from a low-resolution picture of the original art work.

The resulting poor quality image, received by the museum, leads to very poor retrieval accuracy when the fax is used in standard *query by example* searches using, for example, colour or texture matching algorithms. It is obvious that fax images are of very low quality and even the best quality of fax machine cannot prevent the loss of information in the fax image. Special algorithms need to be developed in order to improve the retrieval accuracy for *query by low-quality image* application. Throughout this chapter, fax images will be used as the main source of low-quality image for *query by low-quality image* experiments. Nonetheless some other forms of low-quality image will also be considered.

3.1.1 Analysis of Fax Images

The fax machine is one of the most important communication machines throughout the world. You can walk into nearly any office, big or small, and you will find a fax machine. Connected to a normal phone line, a fax machine allows us to transmit pieces of paper to someone else instantly. Even with the popularity of e-mail, it is nearly impossible to do business without one of these machines today.

During faxing, the scanner in the machine looks at one line of the sheet of paper and looks for a group of black and white spots. It encodes the pattern of spots and sends them through the phone line. The bits for the scanned document travel through the phone line and arrive at a receiving fax machine. The bits are decoded, uncompressed, reassembled into the scanned lines of the original document, and printed onto a paper. However there are many factors that can degrade the quality of the received document from the original. First of all, most fax machines convert colour documents to monochrome documents, and certain colours such as yellow, greenish yellow and light blue are not recognised by the machine. Secondly, if either fax machine is dirty, then the document will be smudged or have black lines though it.

The resolution and contrast setting of the machine is important as well. For most machines, several resolution settings can be chosen such as super fine, fine, standard and half tone. Higher resolution gives better quality but take a longer time to transmit the document, while lower resolution have the opposite effects. The contrast setting also tend to be set high since that will improve the quality of a transferred text document. But for an image document, the received image will tend to be binary-like. Since fax machine operates using telephone lines, noise and distortion can also occur during transmission. Finally the printing quality of the receiving fax machine will also contribute to the quality of the received document. Some fax machines even need to use special paper for printing that tends to turn yellow or brown after a period.

All of these factors may contribute to the poor quality of fax images. Figure 2.2 shows several examples of them. From the figure, it is obvious that the fax images are almost entirely binary (probably because of high contrast setting), and differ significantly from their original images. Figure 3.1 shows the histogram of the fax images compared to the original image. From the histogram, we can see that the pixel distribution of the fax image is pushed towards both ends of the histogram. If the distribution is pushed further towards the two extremes, we will obtain a binary image. The histogram also suggests that no image enhancement or image restoration algorithms could be used to restore the fax image to the original. Therefore if the fax image is to be used as a query for content-based image retrieval, we have to make do with the inadequate information. However, one property of the fax images which has been learned is the fact that they are almost binary in nature, and we could exploit this property for content-based image retrieval applications.

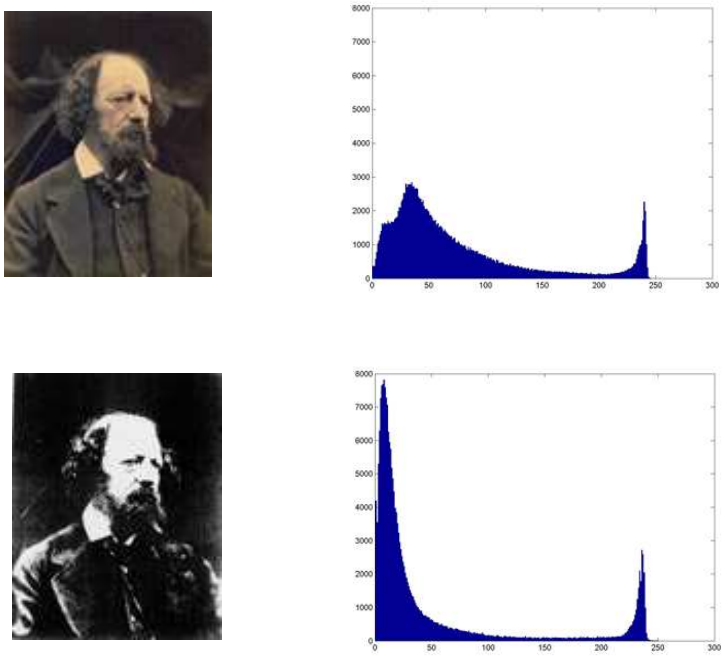


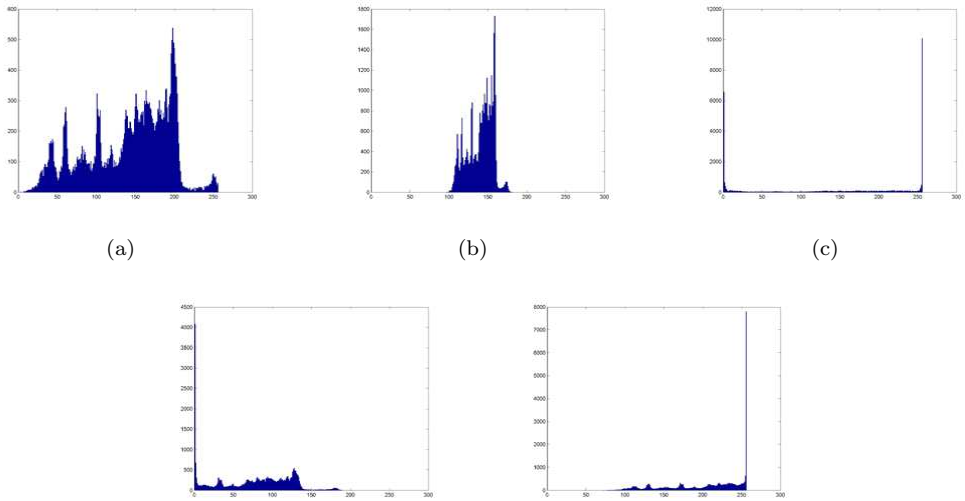
FIGURE 3.1: Histograms of a fax image and its original

3.1.2 Other Low-Quality Image Examples

Besides fax images, there are other forms of low-quality images. This includes images of inappropriate contrast and brightness, highly-compressed images, low resolution images, quantized images, as well as noisy images. All of these forms of low-quality images will also be considered and used as queries for the novel query by low-quality image algorithms to be discussed in the next section.

3.1.2.1 Images of Inappropriate Brightness and Contrast

There are many image acquisition devices available on the market. One of the most popular is the digital camera. But even with a high quality digital camera we can still get a rather poor quality image. For example, an image can be overexposed or underexposed and the resulting images can be either too bright or too dark. The over and underexposed images are also difficult to enhance. When an image is overexposed, its pixels tend to skew toward the high end of the histogram, and simple histogram modification is sometimes inadequate to recover the image. The same can be said about underexposed images. An image can also be of low contrast or high contrast. A low contrast image will have a very narrow histogram, while a high contrast image will have pixels clustered at both the low and high end of the histogram (same as fax images). Figure 3.2 shows histograms of some poor quality monochrome images.



(a) (b) (c)

(d) (e)

FIGURE 3.2: Histogram of (a) 'Correct Image', (b) Low-contrast image, (c) High-contrast image, (d) Dark image, (e) Bright image

3.1.2.2 Highly Compressed Images

In order to save storage space, most digital images are compressed when stored in a computer. The compression technique used to compress the image is usually the lossy compression for standard image collections. This is done by eliminating data that are visually unnecessary and by taking advantage of the redundancy that is inherent in most images. Standard lossy compression techniques can achieve up to 50 times reduction in storage space for still images. However there is a tradeoff in achieving such a high compression ratio. Higher compression means more data being removed, and hence leads to a drop in image quality. When the compression ratio is too high, a significant degradation in image quality is observed. This reduction in quality will be crucial when the image is used as a query in content-based image retrieval applications.

3.1.2.3 Low Resolution Images

Images can have a wide range of resolutions. The higher the resolution, the larger the potential information carried by the image. In the case of a low resolution image, if the resolution is too low, there might not be enough information in the image. The number of pixels determines the resolution of an image, and it is a subjective issue to decide how many pixels are considered to be of low resolution. If these images are used as query in a content-based image retrieval system, problems might arise due to the inadequate information in the image. Various interpolation algorithms are available in order to increase the resolution of the image, but interpolation algorithms do not increase the information in the image. No matter how good the algorithm is, it is impossible to

recreate data that is non-existent in the first place.

3.1.2.4 Quantized Images

The standard number of grey level values used for monochrome images is 256 as studies have shown that the human visual system can differentiate no more than 256 grey levels (97). However, in some applications, images are quantized to some smaller number of grey levels. The smallest number of grey levels for an image is 2, which is the case for binary images. In between these values, users can always set their own number of grey levels for an image, usually a number of the power of 2. However lower grey values mean a reduction in information that can be carried in the image. Using these kinds of images in content-based image retrieval applications can also raise a problem because of the lack of information, as in the previous case.

3.1.2.5 Noisy Images

Images can sometimes be accompanied by noise. Noise is any undesired information that contaminates an image. Noise appears in images from a variety of sources. The digital image acquisition process, which converts an optical image into a continuous electrical signal that is then sampled, is the primary process by which noise appears in digital images. At every step in the process there are fluctuations caused by natural phenomena that add a random value to the exact brightness value for a given pixel. The signal-to-noise ratio determines the amount of noise in an image. The lower the signal-to-noise ratio, the poorer the quality of the image.

3.1.3 Previous Work on *Query by Low-Quality* Image

As far as content-based image retrieval is concerned, there is no comprehensive work on query by low-quality image applications. In most CBIR systems, if a low-quality image is encountered, either as query or database image, they usually enhance the particular image first before processing the information in the image, for example by noise reduction (98; 99; 100) or histogram manipulation (101; 102). Although this may be a good approach in some cases, it is usually effective only on "not so low"-quality images. As we have seen previously, the images we are working on are particularly poor in quality, and enhancing the image will not increase the retrieval accuracy of the system. In the following section, a simple but effective method for dealing with *query by low-quality image* problems is proposed.

3.2 A Novel *Query by Low-Quality Image* (QBLI) Algorithm

Our novel *query by low-quality image* algorithm (103; 104) is based on the wavelet transform. There are several reasons for the use of wavelet transforms in the algorithm. Firstly, the wavelet transform is of low computational intensity, which is an essential property in the proposed algorithm. Secondly, the wavelet transform is known to have very good discrimination when used in image classification, at least for texture retrieval. Finally the feature vector produced by the wavelet transform is compact, that is we can simply take the mean energy of each wavelet channel as a feature. The number of channels of the wavelet transform can be controlled by adjusting the number of decomposition levels during the transform. The compact feature vectors are also an essential property for using the proposed algorithm. These properties make the wavelet transform a suitable image analysis tool for *query by low-quality image* application.

From chapter 2, we learned that there are a number of varieties of wavelet transform. These include the standard or pyramidal wavelet transform (PWT), tree-structured wavelet transform (TWT), discrete wavelet frames (DWF) and Gabor wavelet transform, among others. However, not all of these varieties of wavelet transform have the three properties mentioned above. In fact, the standard wavelet transform is the only technique that offers all three advantages. The tree-structured wavelet transform results in a much higher number of channels, and is quite complex compared to the standard wavelet transform. Moreover, for a non-textured image, the frequency content of an image is concentrated in the low frequency region, thus image decomposition is needed just for the low frequency band. The discrete wavelet frames technique, although it has the same number of channels, is more computationally intensive than the standard wavelet transform. The Gabor wavelet and all other continuous wavelet transforms have an even higher computational complexity. Therefore the PWT will be used in the proposed QBLI algorithm to compute the feature vector of an image.

Now we will discuss how to use the PWT technique in the QBLI algorithm. Since the quality of a low-quality image is so low that it differs substantially from its original, applying the PWT to the original image will not produce a feature vector close enough to the feature vector of the query. To avoid this problem, we exploit the fact that the low-quality image is almost binary in nature. The low-quality image is first converted to a binary image, before the PWT is applied. A similar conversion to binary is applied to each of the database images, choosing a threshold which makes them as close as possible to the binary fax image, before the PWT is applied. Therefore during matching, the comparison is actually made between the binary versions of both the query image and the database images. Using an appropriate thresholds, the binary image of the query should be similar to the binary image of its original. If two binary images are almost the same, their feature vectors should also be similar, hence resulting in correct retrieval

of the original image. One simple approach to compute the threshold value for binary conversion of the database images is to use the percentage of black or white pixels of the query binary. For example, if after binarisation, the percentage of white pixels of the query image is x, then all the database images can be converted to binary in such a way that the percentage of white pixels is also x.

At first, this method seems to be unsuitable for use with the system in Figure 2.1, since in order to compute the feature vector of the database images, the percentage of white pixels is required and that depends on the query images. It is highly ineffective to compute the feature vectors of the database images during retrieval as it makes the response time large, depending on the volume of the database. An effective retrieval system computes the feature vectors of the images in the database in advance, so that only the matching process is needed during retrieval process. Nevertheless, due to the compact nature of the wavelet signatures, we are able to propose an algorithm that suits the system represented in Figure 2.1. The algorithm consists of two steps; *binary image thresholding* and *feature vector computation and comparison*, and these are explained below.

3.2.1 Binary Image Thresholding

As stated earlier, since the query images are almost binary, it is better to compute feature vectors in the binary domain. The query image can be converted to binary by thresholding in the middle of the grey scale range covered by the image. In order for the feature vector of a database image to be compared fairly with the feature vectors from the query, the database image must also be converted to binary. But the choice of threshold is not immediately obvious. For the original database image corresponding to the query, an appropriate threshold is probably one that produces a binary image with the same percentage of black (or white) pixels as the binary form of the query image. We could use this percentage for all the database images, but it varies from query to query. How can the percentage be matched if we are to precompute the feature vectors from the binary versions of all the database images? Note that since the query image is the target and already effectively binary, it is the original database image that must be made as close as possible to the binary query and not vice versa.

One way to solve this problem is to convert each database image into a set of different binary images corresponding to different percentages of black pixels from 0 to 100%. If sufficient binaries are created, the binary query image will then be very similar to one of these binaries for the original image. We propose that 99 binaries are created for each database image corresponding to percentages of black pixels from 1 to 99 in steps of 1% (note that binaries of 0% and 100% are respectively black and white images, and thus are not needed). One might argue that 99 binaries for the database images might be too much or too little. In the experimental section, we will discuss the optimum

number of binaries to be created for each database image. At the moment, for the sake of explanation and as an initial estimate, the simple-to-understand 99 binaries representing 99 threshold percentages will be used.

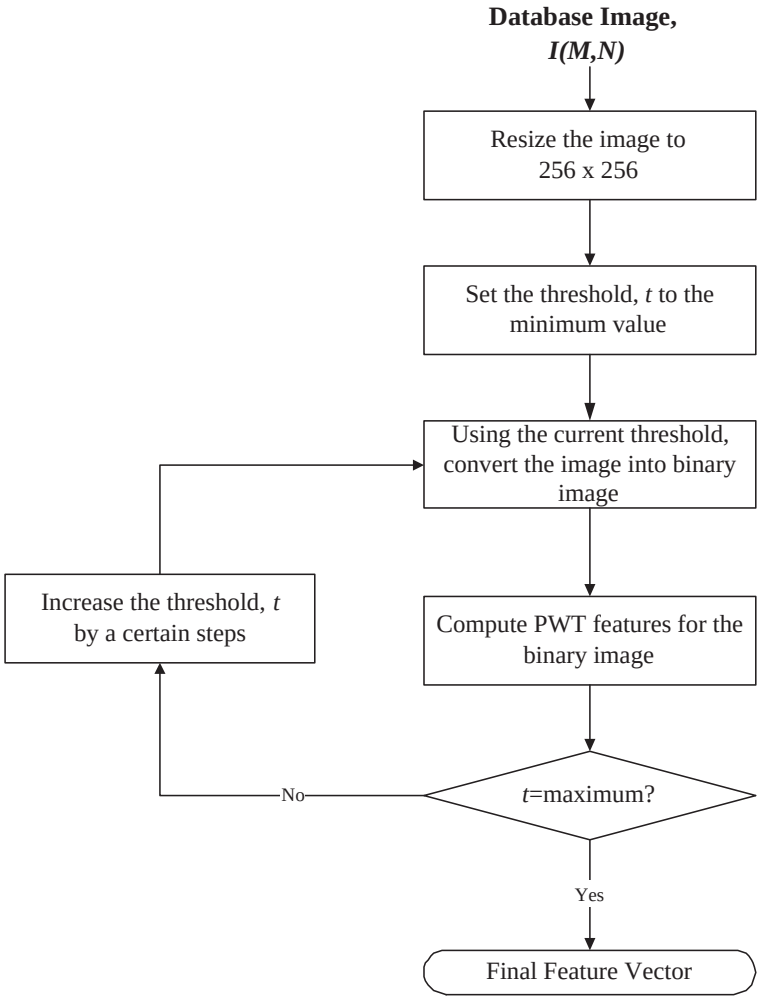


FIGURE 3.3: Flowchart of the proposed algorithm for database feature extraction

However, the binaries do not need to be stored. Calculating the feature vectors for the database involves calculating the PWT for each of the binary images for each image in the database. This is implementable since the PWT is a fast algorithm and, more importantly, the feature vectors for each binary image have only a relatively small number of coefficients. There will be 99 sets of feature vectors for each database image, but during matching, once the percentage of pixels of the query is known, only one of these 99 sets will be used for comparison, namely the set associated with the same pixel percentage as the binary query image. Figures 3.3 and 3.4 show the flowchart of the proposed algorithm for the feature extraction stage and the retrieval stage respectively.

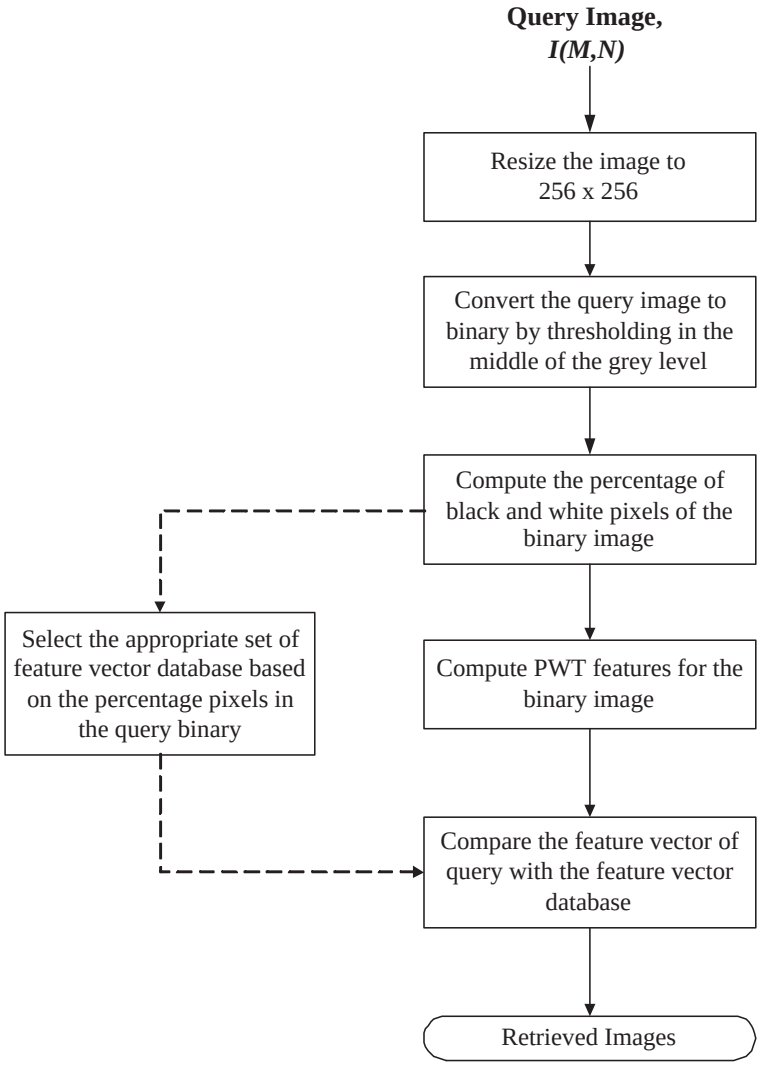


FIGURE 3.4: Flowchart of the proposed algorithm for retrieval stage

3.2.2 Feature Vector Computation and Comparison

As described earlier, the PWT is used as a tool for feature vector extraction. Since the PWT is usually applied on dyadic square images, the binary images are all resized to 256×256. Other image sizes can also be used although bigger sizes will increase the computational load while smaller sizes will reduce the information within the image. The 256×256 size is thus the most suitable option. For simplicity, nearest neighbour interpolation and decimation will be used for the resizing process. The resizing can also be done before the binary conversion. The PWT algorithm is applied and the image is decomposed into four sub-images (LL,LH,HL and HH, refer chapter 2). The LL band is decomposed further until the desired number of decomposition levels is achieved. For a start, we set the smallest sub-images (sub-images corresponding to the highest level) are of size 4×4, which means six levels of decomposition for an image of size 256×256. This results in 19 different sub-images or sub-bands.

Once the wavelet coefficients of a binary image are available, features are computed from each sub-band, resulting in 19 features for each binary image. The mean μ is the energy measure used to compute the features. Let the image sub-band be $W_{mn}(x, y)$ while mn denotes the specific sub-band, m is the decomposition level and *n=1,2,3,4* indicates the *LL, LH, HL, HH* bands respectively, then μ_{mn} is calculated by:

$$\mu_{mn} = \frac{1}{N_{mn}^2} \sum_y \sum_x |W_{mn}(x, y)| \tag{3.1}$$

where N is the length of a particular sub-band mn.There are several other energy measures that can be used as the feature such as the standard deviation, the number of zero-crossings etc., and these are evaluated in the context of texture retrieval in chapter 4 of this thesis. But in order to keep the algorithm simple, only the mean energy is used. Incorporating any other energy measure with the mean energy will increase the total number of features for a particular database image quite drastically, and hence is not very suitable for the proposed algorithm. The feature vector f for a particular binary image is therefore:

$$\begin{aligned} f &= [\mu_{mn}]\,,\ n \neq 1 \text{ except for the coarsest level, i.e. } m=6, \\ &= [\mu_{12}, \mu_{13}, \mu_{14}, \mu_{22}, ...\mu_{61}, \mu_{62}, \mu_{63}, \mu_{64}] \end{aligned} \tag{3.2}$$

The feature vectors for the database images will have 99×19=1881 coefficients, although only 19 will be used for comparison in each retrieval task. During matching, a distance classifier is used to compute the similarity between the query and each database images. There are quite a few distance classifiers available and their performance for this particular algorithm will also be investigated. However as a default, the widely known minimum Euclidean distance will be used for the early part of the experiments. The minimum Euclidean distance between 2 features i and j is given by:

$$d_{Euclidean}(i, j) = \sum_m \sum_n \left[\mu_{mn}^{(i)} - \mu_{mn}^{(j)}\right]^2 \tag{3.3}$$

The square root is omitted for computational efficiency as it does not affect the order of similarity. Once the distances are computed, the images will be retrieved in order of increasing distance from the query image.

3.3 Available Methods for Comparison

As previously mentioned, there is little published research specifically on query by low-quality images available in the literature. Thus it is difficult to compare the performance of the proposed algorithm with other published algorithms. However, in this section we

listed two techniques which will be compared against the proposed algorithm. One of the techniques is a simple pixel matching algorithm which we develop for comparison with the proposed QBLI method. The other technique is simply to use the wavelet transform to extract features from the low-quality image without any binarisation processes. It is intended to show the superiority of the inclusion of the binarisation in solving the particular retrieval problem.

3.3.1 Pixel Matching Algorithm

The pixel matching algorithm is almost identical to the proposed QBLI algorithm except that instead of using PWT coefficients of binary images as features, it uses pixel by pixel matching between the binary query and the database queries. It is intended to show the effect of using PWT coefficients as features in the QBLI algorithm. For a particular database image, 99 binaries are created and stored as features. During retrieval, the query is converted to binary and the percentage of black pixels is computed. A pixel by pixel matching is then performed between the query binary and the database binary corresponding to the percentage of black pixels. The number of matching pixels, as a percentage of the total number of pixels, is used as the similarity measure and the database images are then retrieved in decreasing order of similarity.

This method however uses large amount of storage. A 256×256 image requires $\frac{256 \times 256 \times 99}{8bit}$ $\approx 800kB$ of memory to store all the features. An alternative way to perform the pixel matching algorithm is by performing binary conversion during the retrieval process. The query image is converted to be totally binary, and the percentage of black pixels is computed. The database image to be compared to the query is then converted to binary in such a way that the percentage of its black pixels is the same as in the binary query image. Both binaries are re-sampled to the same size and a pixel by pixel comparison is made. Performing the pixel matching algorithm this way, there is no need for feature storage at all, although the computational load involved during interactive retrieval is very high.

Based on the description above, the pixel matching algorithm is not appropriate for interactive retrievals since it either involves a large storage or a very slow interactive retrieval. However, as will be shown later, this method does give a very high accuracy in retrieving images based on low-quality query. This method is used as a yardstick for the evaluation of the proposed wavelet-based QBLI algorithm since it gives such good retrieval accuracy. In the experiment section, the second approach to pixel matching algorithm will be used because there is not much different in computational speed with the increasing size of re-sampled images, as oppose to linear increase in storage space for the first approach.

3.3.2 Pyramidal Wavelet Transform

Since the proposed QBLI method uses the pyramidal wavelet transform as the feature extractor, it is reasonable that the PWT algorithm is compared to the novel algorithm. By comparing it with the feature extractor itself, we can evaluate the true effectiveness of the binarisation stage in the QBLI. We can also observe how the retrieval accuracy would be if we are using standard feature extraction techniques without considering the high difference between the low-quality image and its original.

The procedure for the retrieval using the pyramidal wavelet transform is relatively straight forward. The entire database image is first resized to a dyadic square image. In this experiment, we use the same size as in the proposed QBLI method, which is 256×256. The pyramidal wavelet transform is applied to the database image, and 19 channels are produced. The mean energy (as in Equation 3.1) is computed from each channel. During the retrieval process, the query image is also resized to 256×256 and features are computed in the same way as for the database image. The similarity between the query and database image is computed using the function in Equation 3.3, and the images are retrieved in order of increasing distance.

3.4 Experimental Analysis

The database used in our experiment consists of 1062 images of various types and sizes, taken from the Victoria and Albert museum collection. From the 1062, 20 images are selected as target images in the experiments, and are shown in Figure 3.5. As mentioned before, fax images are the primary sources for the low-quality image in this paper, hence a genuine fax version of the 20 selected images are used as queries for the retrieval experiment. The 20 fax images are shown in Figure 3.6. The evaluation is based on the ability of the algorithm to retrieve the original image when the fax version of the original is used as the query. Experiments conducted include the evaluation of the effectiveness of the proposed algorithm, the performance using different distance classifiers, as well as different numbers of decomposition levels and wavelet bases, investigation of the optimum number of binaries to be created, and finally the performance of the algorithm on other types of low-quality images.

3.4.1 Evaluation of the Novel Algorithm

In this particular experiment, the number of binaries created for each database image is set to 99, representing 99 different percentages of black pixels. The wavelet basis used for wavelet transform decomposition is the Daubechies 8-tap wavelet. The distance metric used is the minimum Euclidean distance, and the number of decomposition levels

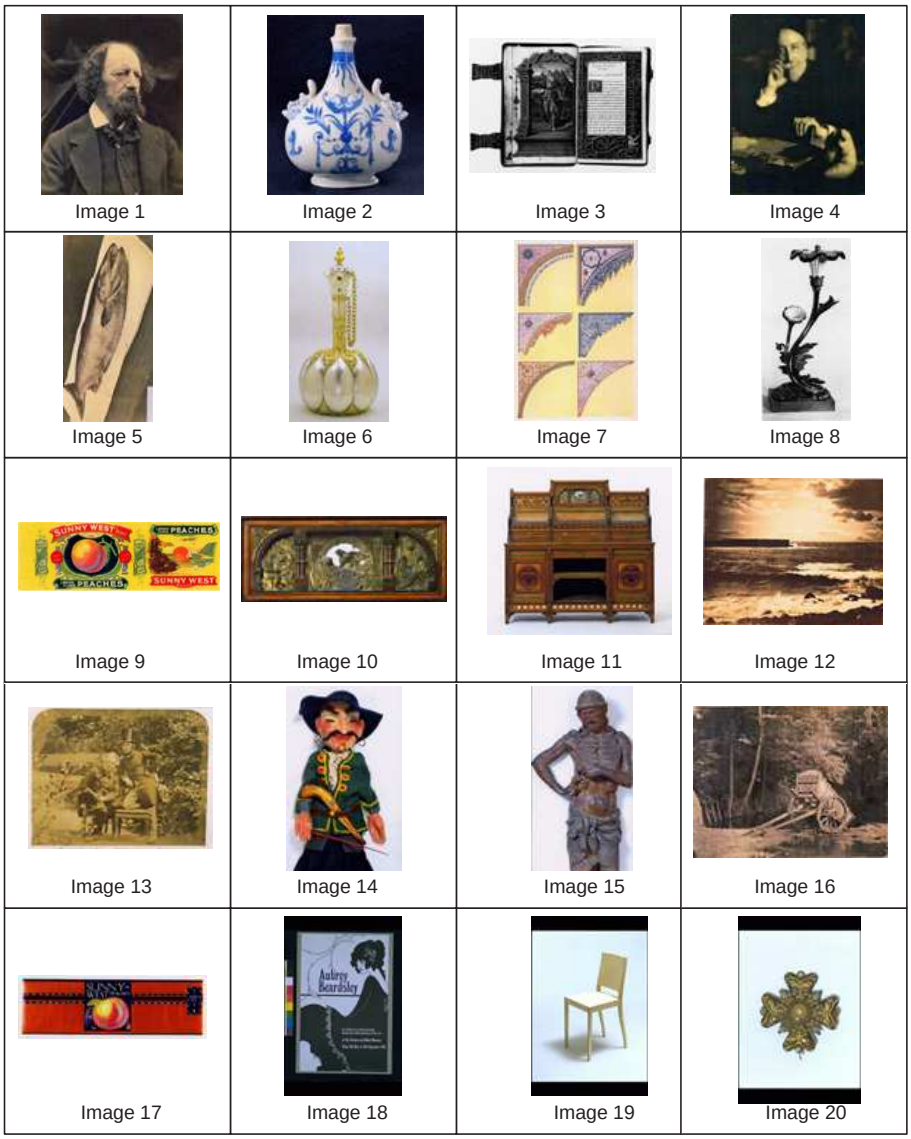


FIGURE 3.5: 20 selected images from the database

is set to six, which means 19 features for each binary image. The results for the novel QBLI algorithm are shown in table 3.1, where the retrieval position of the original image among the 1062 database images is shown. The table also shows the results obtained by using a basic *query by example* retrieval using PWT features calculated from the raw query and database images without the initial binarisation stage.

It can be seen that the basic PWT *query by example* algorithm is particularly poor for fax queries, but the retrieval results obtained using the QBLI algorithm are very encouraging. All the original images are retrieved within the top 5. This is a good result considering the poor quality of the fax images and the reasonably large image database used. The result suggests the importance of the binarisation process in the algorithm. It also suggests that the distances between the fax images and their originals are very close and should still produce good results for a larger image database. Figure

FIGURE 3.6: 20 fax images corresponding to the images in Figure 3.5

3.7 shows an example of three retrieval results using the QBLI algorithm.

Table 3.1 also shows the results using the pixel matching algorithm. The result was obtained using the second approach to pixel matching algorithm with the re-sampled size set to 256×256. As expected, the pixel matching algorithm gives very good retrieval results. All the originals were returned as the first match, except for one case only, which fails because of the poor state around the edges of that particular query image, which leads to inaccurate computation of the percentage of black pixels. The result of the QBLI algorithm for that particular query is however very promising. This shows that the pixel matching algorithm is very sensitive to noise, while the QBLI algorithm is much more robust. It was also observed that the size of the re-sampled image is quite critical for pixel matching algorithm. The bigger the re-sampled image, the better the accuracy, with the best result was observed when the re-sampled images are set to 256×256 or

Query Image No.	Rank of Original		
	Basic PWT	Pixel Matching	Novel QBLI
1	104	1	1
2	369	1	1
3	15	1	1
4	21	1	3
5	272	1	1
6	130	1	1
7	258	1	1
8	2	1	3
9	502	1	1
10	302	15	2

Query Image No.	Rank of Original		
	Basic PWT	Pixel Matching	Novel QBLI
11	603	1	1
12	299	1	1
13	60	1	1
14	495	1	4
15	500	1	2
16	339	1	1
17	15	1	2
18	264	1	4
19	1	1	1
20	1	1	1

TABLE 3.1: Retrieval results using 20 fax images on a database of 1062 images

Time taken to retrieve images (in seconds)	Basic PWT Technique	Pixel Matching Technique	Novel QBLI Technique
	1	130	1

TABLE 3.2: Comparison of speed between the three algorithms

bigger.

Table 3.2 compares the average time taken for retrieving images from the database of 1062 images using the three techniques. The times are observed on a 700 MHz Xeon processor. From table 3.1 and table 3.2 it can be seen that the proposed QBLI algorithm almost equals the pixel matching algorithm in terms of retrieval performance, but involves a much smaller computational load, and therefore is much better for interactive queries. It is also important to point out the fact that the PWT algorithm applied on binary images helps in minimising computation time. Logically, a computation based on just ones and zeros is much quicker than a computation based on multiple values. To sum up, it can be said that the fast QBLI method integrates the high accuracy of the pixel matching method with the low computational load and the robustness of the basic PWT method.

3.4.2 The Effect of the Distance Metric

In the previous experiment, we used the minimum Euclidean distance as distance classifier for the proposed algorithm. There are however several other distance metrics that can be used with the algorithm such as the Manhattan distance, Bayesian distance and Mahalanobis distance. However with the exception of Manhattan distance, these distance metrics are rather complex in nature. For the sake of computational simplicity, we restrict our distance metrics only to simple functions, which means only the Euclidean

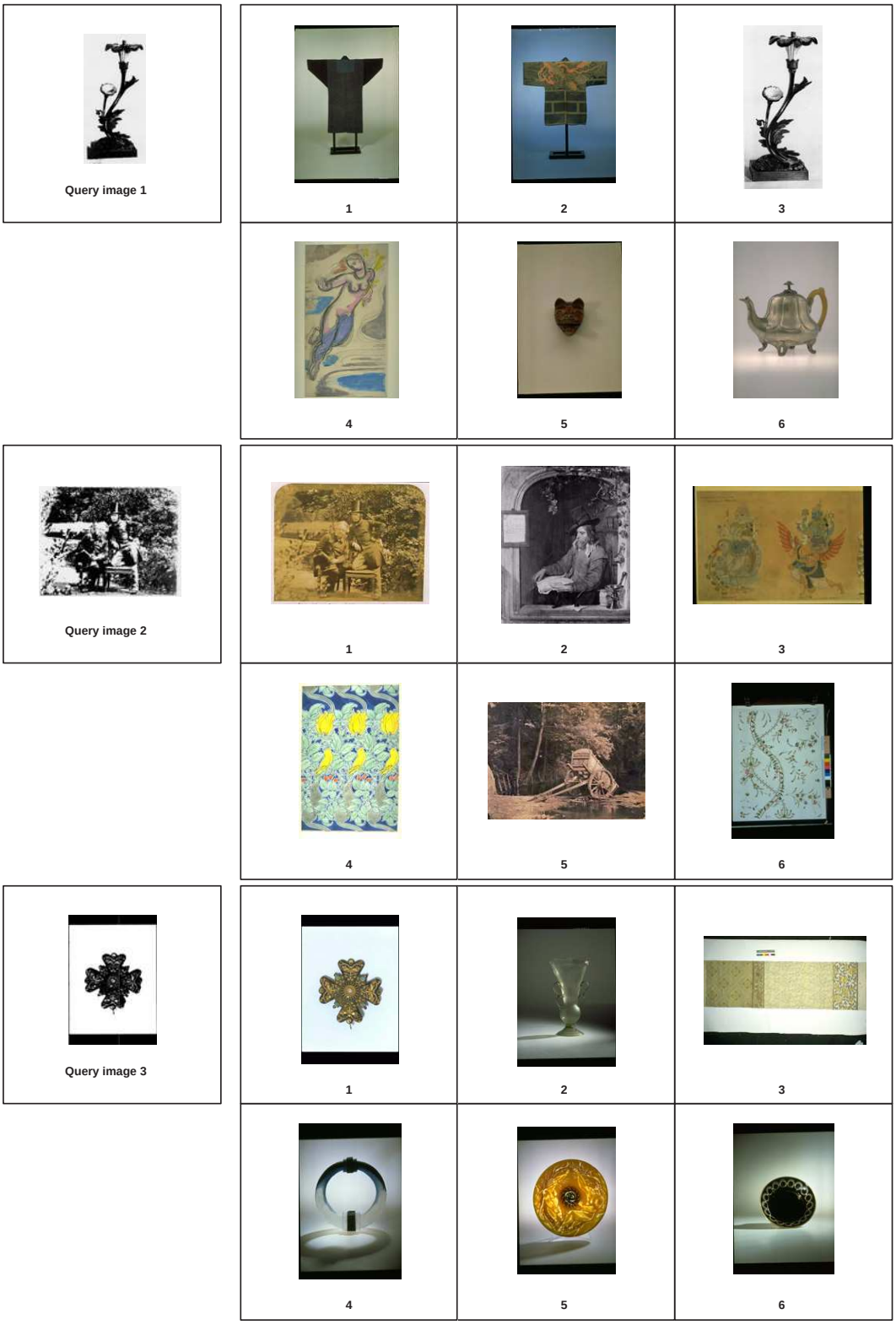


FIGURE 3.7: Fax images and their top six retrieved images.

Query Image	Rank of Original		
	Manh.	Norm. Eucl.	Norm. Manh.
1	1	1	2
2	1	1	1
3	1	1	4
4	3	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	2	1	1
9	1	1	1
10	2	1	1

Query Image	Rank of Original		
	Manh.	Norm. Eucl.	Norm. Manh.
11	1	1	1
12	1	1	1
13	1	1	1
14	3	5	2
15	2	1	3
16	1	1	1
17	1	1	1
18	3	61	61
19	1	1	1
20	1	1	1

TABLE 3.3: Retrieval results using different distance metrics

and Manhattan distance will be tested. However we will also introduce normalisation in distance computation to observe if the addition of normalisation can further boost the retrieval performance. The idea of normalising the feature vectors arises due to different range of energy values between different wavelet channels, and it is feared that this might lead to contributions from some channels become insignificant. In the following experiment, we will observe the performance of three different distance classifiers, namely the Manhattan distance, the normalised Euclidean distance and the normalised Manhattan distance, compared to the Euclidean distance. The three distance metrics are given below:

$$d_{Manhattan}(i,j) = \sum_m \sum_n \left| \mu_{mn}^{(i)} - \mu_{mn}^{(j)} \right| \tag{3.4}$$

$$d_{Norm.Euclidean}(i,j) = \sum_m \sum_n \left[\frac{\mu_{mn}^{(i)} - \mu_{mn}^{(j)}}{\sigma_{mn}} \right]^2 \tag{3.5}$$

$$d_{Norm.Manhattan}(i,j) = \sum_m \sum_n \left| \frac{\mu_{mn}^{(i)} - \mu_{mn}^{(j)}}{\sigma_{mn}} \right| \tag{3.6}$$

where σ_{mn} is the standard deviation of a particular feature over the entire database.

Table 3.3 shows the ranking of the retrieved results using three different distance metrics. From the table, there is not much difference between the three distance metrics compared to the Euclidean distance (refer Table 3.1, third column). However closer inspection suggests that the standard Euclidean and Manhattan distance metrics give somewhat more stable retrieval results. This is because of the ranking of fax image 18, where in both the normalised cases, the ranking is outside the top 60. Careful inspection shows that fax image 18 has noise distributed across it in terms of black dots on a brighter background. After binarisation, it appears that the noise is still present in the binary image. This noise is usually of high frequency, thus it will be captured by the high frequency channels of the wavelet decomposition.

The high frequency channels of the wavelet decomposition have a small range of energy values compared to the other channels. By normalising the features, the noise in the high frequency channels is amplified and therefore results in high dissimilarity between the fax and the original. Without the normalisation process, the dissimilarity measure will not be too big because the high frequency channels contribute only a small portion of the overall measurement. Since low quality images are very likely to contain noise as in fax image 18, it is better not to normalise the features. Further inspection shows that although the features have a wide range of energy value, none of the individual feature differences really dominate in a way that a particular feature difference is completely superior to the others. Hence there really is no need for normalisation for the dissimilarity computation.

Between the non-normalised Euclidean and Manhattan distance, the Manhattan distance appears to be better than the Euclidean distance, where all the retrieved targets are within the top 3. The reason for this is that with Euclidean distance, the individual feature difference is squared, which further reduces the contribution of smaller feature differences. By using Manhattan distance, the individual feature differences are added together which preserves the contribution of each individual feature. Using this metric, none of the feature differences are amplified or neglected, hence resulting in a better retrieval performance. The Manhattan distance will therefore be used as the distance classifier for the QBLI algorithm.

3.4.3 The Effect of Different Numbers of Decomposition Levels, L

The number of decomposition levels is an important factor in the wavelet transform. It decides how many frequency channels a particular transform will have. The bigger the number of decomposition levels, the more the detail in the image is captured. However since the pyramidal wavelet transform decimates its output by a factor of 2, if the decomposition is continued all the way, we will obtain channels with only 1 pixel resolution. This might lead to inaccurate readings and unstable features, and will increase the number of features to be computed. Therefore we will have to limit the smallest resolution for a particular channel, and it was found experimentally that 4×4 pixels is the smallest appropriate channel resolution. For a 256×256 image, this leads to a maximum of 6 levels of decomposition, the one used in previous experiments.

However we might want to use fewer decomposition levels since it involves fewer computations and fewer features. Table 3.4 shows the results of the retrieval experiments using different numbers of decomposition levels using Manhattan distance. From the table, clearly the retrieval performance decreases with the decrease of the number of decomposition levels. The decrease in performance is quite drastic for some query images, while steady in most other queries, depending on the quality of the queries. However since using six levels of decomposition gives the best performance, and that its corresponding

Query Image	Rank of Original			
	L=6	L=5	L=4	L=3
1	1	2	5	11
2	1	1	1	1
3	1	3	19	106
4	3	1	2	1
5	1	1	3	18
6	1	9	5	1
7	1	1	1	2
8	2	2	1	5
9	1	1	1	2
10	2	1	14	2

Query Image	Rank of Original			
	L=6	L=5	L=4	L=3
11	1	1	2	83
12	1	1	3	5
13	1	1	1	2
14	3	57	143	387
15	2	6	3	8
16	1	1	2	6
17	1	1	1	1
18	3	16	41	178
19	1	1	2	53
20	1	1	1	2

TABLE 3.4: Retrieval results using different numbers of decomposition levels

Haar	Daubechies 4-tap	Daubechies 8-tap	Coiflet 6-tap	Symmlet 8-tap	Binary WT	
					lowpass	highpass
-0.7071	-0.4830	-0.2304	0.0727	-0.0322	1	1
0.7071	0.8365	0.7148	0.3379	-0.0126	1	1
	-0.2241	-0.6309	-0.8526	0.0992	1	1
	-0.1294	-0.0280	0.3849	0.2979	0	1
		0.1870	0.0727	-0.8037	1	1
		0.0308	-0.0157	0.4976	0	1
		-0.0329		0.0296	1	0
		-0.0106		-0.0758	0	0

TABLE 3.5: Filter coefficients of different wavelet basis

number of features (19 features) is still compact, we adapt these parameters for the QBLI algorithm.

3.4.4 The Effect of Using Different Wavelet Bases

In the previous experiments, the Daubechies 8-tap wavelet is used as the wavelet basis for decomposition. Four more wavelet bases are tested in order to observe the effect of using different wavelet bases in the algorithm. In addition, a special binary wavelet transform decomposition is also tested since it is interesting to see if total binary operation on binary images might give better results. Six decomposition levels are applied for each case, and the Manhattan distance is used as classifier. Table 3.6 shows the retrieval results using the Daubechies 4-tap wavelet, Coiflet 6-tap wavelet, Symmlet 8-tap wavelet, Haar wavelet and the binary wavelet transform. Table 3.5 shows the filter coefficients of each wavelet basis. The bases for binary wavelet transform are the ones suggested by Swanson and Tewfik (105).

From the results table, the binary wavelet transform particularly gives a very poor result. The other normal wavelet transform provides comparable performance. The poor retrieval results for the binary wavelet transform are due to the lack of information carried by the output of the binary wavelet transform. The binary wavelet transform

Query Image	Rank of Original						Query Image	Rank of Original					
	D4	D8	C6	S8	H	B		D4	D8	C6	S8	H	B
1	1	2	3	1	1	184	11	1	1	1	1	1	335
2	1	2	1	1	2	26	12	1	1	1	1	1	84
3	1	2	1	1	2	302	13	1	2	3	1	1	53
4	3	1	1	1	1	19	14	4	38	1	7	10	605
5	1	1	1	1	1	228	15	2	1	1	1	1	448
6	1	1	1	1	1	5	16	1	2	1	1	2	38
7	1	1	1	1	1	1	17	2	1	1	1	1	1
8	3	3	1	1	1	3	18	4	59	1	1	9	248
9	1	1	1	1	1	168	19	1	1	1	1	1	301
10	2	29	1	1	8	104	20	1	1	1	1	1	1

TABLE 3.6: Retrieval results using different wavelet bases (D4=Daubechies 4-tap, D8=Daubechies 8-tap, C6=Coiflet 6-tap, S8=Symmlet 8-tap, H=Haar, and B=binary wavelet transform

performs a special filtering operation which results in a binary output. Since we are using mean energy as the feature, computing this feature on binary channels does not provide enough information, which leads to poor discrimination between images. While the binary wavelet transform is good in the compression of binary images, it is not suitable for image retrieval.

The other four wavelet bases give a good performance, especially the Coiflet and Symmlet bases. It can be concluded that the choice of wavelet bases is not very crucial in the QBLI algorithm, although based on this particular test, the Coiflet wavelet basis gives the best result and will be used as the basis for the algorithm. One might argue that the Haar wavelet should give better performance because of the binary-like nature of its filter. This might be true, but since we are dealing with low quality images, in which the query binary does not really resemble the target binary completely due to noise etc., a bit of tolerance is needed in computing the features. In this case, the Haar wavelet may be too accurate in representing the binary images so that when noise is present, the dissimilarity measure increases dramatically. The Daubechies 8-tap, Coiflet 6-tap and Symmlet 8-tap wavelets although they may not be very accurate in representing binary images, they are better for the reasons above.

3.4.5 Optimum Threshold for Binarisation

It is debatable whether the choice of creating 99 different binaries for each database image is optimal. Different numbers of binaries are tested for the algorithm to find the optimal number of binaries to be created. Steps of 0.5%, 2%, 5% and 10% (which creates 199, 49, 19 and 9 binaries respectively) are tested and the results are shown in Table 3.7. Note that there is quite a large reduction in the number of features to be stored by reducing the number of binaries to be created. Experiments are conducted using Coiflet 6-tap wavelet bases, six decomposition levels and Manhattan distance as classifier.

Query Image	Rank of Original					Query Image	Rank of Original				
	0.5%	1%	2%	5%	10%		0.5%	1%	2%	5%	10%
1	3	3	6	3	155	11	1	1	1	1	1
2	1	1	6	23	9	12	1	1	1	1	1
3	1	1	1	1	2	13	3	3	3	4	5
4	1	1	1	18	55	14	1	1	1	1	1
5	1	1	2	1	7	15	1	1	1	1	1
6	1	1	1	1	1	16	1	1	1	1	1
7	1	1	1	1	1	17	1	1	12	167	146
8	1	1	1	1	3	18	1	1	2	2	2
9	1	1	1	1	3	19	1	1	1	5	58
10	1	1	1	3	3	20	1	1	1	1	1

TABLE 3.7: Retrieval results using different number of binaries

From the table, increasing the number of binaries from 99 to 199 does not appear to improve the retrieval performance. For each case of the query image, the same result using only 99 binaries are observed. Hence there is no need for increasing the numbers. The performance of using 49 binaries does not differ much from that of using 99 binaries, except for some specific queries. All the retrieved targets are still within the top 15, and the fact that it halves the number of features to be stored might make 49 binaries useful. As the number of binaries decrease to 19 and 9, the performance continued to drop, which suggest that the steps of 5% and 10% might be too big and results in poor binary matching.

In actuality, as long as the query binaries resemble closely enough the target binary, we will obtain good results. The problem arises when the percentage of pixels of the binary queries appear in the middle of two target binaries, say 85% of query binary compared to 80% and 90% of target binaries which can lead to a quite different binary matching. In this case, the 5% and 10% steps proved to be too big a gap for some queries and therefore should not be used. This experiment also suggests that in order to get the best possible result, we only have to use a maximum of 99 binaries as increasing them does not improve the performance. The choice of optimum number of binaries is therefore a tradeoff between retrieval performance and the computational speed. If the speed constraint is not very crucial, using 99 binaries is better, otherwise using 49 binaries offer a comparable alternative. In our case, we chose to use 99 binaries for the QBLI algorithm.

3.4.6 Using Other Forms of Low-quality Image as Query

We now consider other types of low-quality images besides fax images to establish whether the proposed algorithm works for these particular types of images. As mentioned previously, we have listed 5 other low-quality image types which are the image of inappropriate brightness and contrast, highly compressed images, low resolution images,

FIGURE 3.8: Target image (top left) and its five modifications on brightness/contrast of an image.

quantized images and noisy images. The same twenty images as in Figure 3.5 are used as the target for retrieval with the query being the modified low-quality version of them. All experiments are conducted using 99 binaries for each image, Coiflet 6-tap wavelet basis, having six decomposition levels and utilizing Manhattan distance as distance classifier, as this combination is found to be the best in the previous section.

3.4.6.1 Images of Inappropriate Brightness and Contrast

The twenty selected images are modified by varying their brightness and contrast in order to make 20 queries for the experiment. We have conducted 5 different experiments using 5 different brightness/contrast modifications. The modifications are high brightness (75% increase in original brightness), low brightness (-75% of original brightness), high contrast (75% of original contrast), low contrast (-75% of original contrast), and random combinations. Figure 3.8 shows an example of query images corresponding to the 5 modifications above. The target image of the query is also shown for comparison in quality.

Experimental results show that for all queries, the target images are retrieved as the first rank. Clearly from the result the proposed algorithm works well with this kind of low-quality images. Note that the queries are generated by performing some operation on the histogram of its originals. As long as the operation does not entirely change the histogram, the proposed QBLI algorithm lends itself well in solving this type of problem. Problems will only arise when a non-linear operation is performed on the original resulting in the pixel distribution of the image being completely modified, in which case the modified image will probably be very different from its original.

FIGURE 3.9: Target image (top left) and its corresponding highly compressed version.

3.4.6.2 Highly Compressed Images

Five different compression quality are used to obtain low-quality queries. Each of these queries are produced using MATLAB, and are compressed using the standard JPEG compression. The quality setting in JPEG compression is measured by values between 0 and 100, with 0 corresponding to the highest compression (worst quality) and 100 corresponding to the lowest (best quality). The default quality setting for JPEG compression is 75. In this experiment, the five different compression qualities take the values of 0, 5, 10, 20 and 30. Figure 3.9 shows an example of a target image and the corresponding five highly compressed versions of it.

Table 3.8 shows the retrieval results for the highly compressed images. From the table, it is observed that the proposed algorithm works well using this kind of low-quality images, even when using the lowest compression quality as query. The queries having compression quality of 0 and 5 in particular are very poor, yet the algorithm still manages to achieve very promising retrieval. As the compression quality increases, the performance increases unsurprisingly because the better the quality, the more it resembles the queries' originals.

3.4.6.3 Low Resolution Images

The twenty selected images are resized to a lower resolution to produce low-quality images of this type. However since in the algorithm we resized all images to 256×256, the twenty selected images need to be resized to a resolution smaller than 256, otherwise there will not be the same amount of information between the queries and the targets,

Query Image	Rank of Original				
	Q0	Q5	Q10	Q20	Q30
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	1
7	3	1	1	1	1
8	1	1	1	1	1
9	1	1	1	1	1
10	1	1	1	1	1

Query Image	Rank of Original				
	Q0	Q5	Q10	Q20	Q30
11	1	1	1	1	1
12	1	1	1	1	1
13	1	1	1	1	1
14	1	1	1	1	1
15	1	1	1	1	1
16	2	1	1	1	1
17	1	1	1	1	1
18	1	1	1	1	1
19	3	1	1	1	1
20	1	1	1	1	1

TABLE 3.8: Retrieval results for highly compressed images, QX represent quality measure.

Query Image	Rank of Original		
	$r = 32$	$r = 64$	$r = 128$
1	3	1	1
2	1	1	1
3	4	4	1
4	1	1	1
5	2	1	1
6	4	1	1
7	10	2	1
8	1	1	1
9	398	18	1
10	487	4	1

Query Image	Rank of Original		
	$r = 32$	$r = 64$	$r = 128$
11	3	1	1
12	1	1	1
13	28	3	1
14	1	1	1
15	1	1	1
16	45	1	1
17	632	134	2
18	39	2	1
19	262	79	13
20	9	3	2

TABLE 3.9: Retrieval results for low-resolution images, where r is the resolution of image's longest dimension.

and the queries cannot be considered as low-quality images. We choose to resize the selected images into having 32, 64 and 128 pixels along their longest dimension. Table 3.9 shows the retrieval results for the low-resolution queries.

The result of resolution 32 is quite poor compared to the other resolutions. This is understandable since an image having 32 pixels in its longest dimension is actually a very small image and not much information is carried. We also have to notice that query images 9, 10 and 17 have a high length to width ratio. If their longest dimension is 32, their other dimension will have far less pixels resulting in a very small amount of information along that particular dimension. Increasing the resolution means increasing the amount of information carried, hence the reason for the much better results for resolution 64 and 128. Apart from the very small resolution of query images 9, 10 and 17 which is responsible for the very poor retrieval results, the other queries give quite a good performance, although not as good as the 2 previous types of low-quality images. Hence the proposed algorithm is also suitable in solving this type of low-quality image.

FIGURE 3.10: Target image (top left) and its quantized version.

3.4.6.4 Quantized Images

Four different quantized images are produced, having a quantisation of 2 (binary), 4, 8 and 16 grey levels. In all cases, the quantized images are poor compared to the originals. Figure 3.10 shows an example of a target image and its corresponding 2, 4, 8, and 16 grey levels quantized version.

Experimental results show that all target images are retrieved as the first match, hence the performances of all quantized cases are very good. The binary queries (2 levels of grey values) are in fact not much different from the fax images, and since the proposed algorithm operates in binary, it lends itself very well with the binary queries. It is also worth mentioning that the performance of the other three quantized images are exactly the same as with the binary cases, because no matter how many grey level values we use, they are all going to be converted to binaries, hence will also work well with the QBLI algorithm. We can conclude that the proposed algorithm is very well suited to this type of low-quality images.

3.4.6.5 Noisy Images

The 20 selected images shown in Figure 3.5 have noise added to produce noisy image queries. Gaussian noise having zero mean with three different strengths are chosen. The strength of the noise is characterized by their standard deviation, and values of $\sigma =$ 0.005, 0.01 and 0.02 were used. In addition a second type of noise, in terms of image blurring, will also be tested. Pixel averaging of radius five pixels are performed on the selected images to produce blur images. Figure 3.10 shows an example of a target image

FIGURE 3.11: Target image (top left) and its noisy version.

Query Image	Rank of Original			
	Gaussian, σ=			Blurred images
	0.005	0.01	0.02	
1	1	1	12	1
2	1	1	1	1
3	1	1	1	7
4	1	1	1	1
5	1	1	1	1
6	1	1	4	1
7	1	1	1	1
8	1	1	1	1
9	2	3	20	335
10	1	2	41	6

Query Image	Rank of Original			
	Gaussian, σ=			Blurred images
	0.005	0.01	0.02	
11	1	1	1	1
12	1	1	1	1
13	4	14	76	3
14	1	1	1	1
15	1	1	1	1
16	1	1	8	45
17	1	1	17	308
18	1	1	1	44
19	8	1	7	4
20	1	1	1	2

TABLE 3.10: Retrieval results for noisy images

and its four noisy image versions.

Table 3.10 shows the result of the retrieval experiment. For the Gaussian noise, the retrieval rate are not significantly affected for $\sigma = 0.005$. As the strength of the noise increases, the performance of the algorithm decreases, although they are still giving acceptable retrieval results. This shows that the proposed algorithm is also suitable in tackling noise related problems. As for the blurred images, except for a few queries, the proposed algorithm also works well to retrieve the target images. The few poor results are for queries 9 and 17, and careful inspection shows that the particular blurred images are quite distorted compared to their originals.

3.5 Chapter Summary

In this chapter, a novel *query by low-quality image* technique which uses a binary matching algorithm together with pyramidal wavelet transform is proposed. A simple but highly time consuming technique, the pixel matching technique is also developed, to be use as a yardstick for the evaluation of the proposed QBLI algorithm. The pixel matching technique gave very good retrieval accuracy, while the proposed QBLI algorithm gave a comparable performance. This implies that the QBLI algorithm is almost as good as the pixel matching algorithm, but has the advantage of being much faster computationally, making it more suitable for interactive use. The novel QBLI method illustrates the importance of the wavelet-based feature extractor in image analysis, where in PWT, we have a very fast algorithm and compact feature vectors. These two constraints are important in using the proposed QBLI method where multiple feature extractions as well as multiple feature vectors are necessary for each database image. Other feature extraction techniques are either slow or have large feature vectors.

Another important observation from the experiments conducted is that the binary image matching and thresholding method proposed is an excellent way to search using low-quality images. This is shown by the fact that the basic PWT technique gives a very poor retrieval accuracy when used for this kind of problem. This suggests that other general purpose feature extraction methods might also fail to deliver a good performance if not accompanied by a special pre-process like the binary image thresholding and matching technique. Further experiments suggested the use of Manhattan distance as the distance classifier in order to get the best results. Normalising the distance function appears to over-amplify the noise associated with the queries and therefore is not suggested. The choice of decomposition levels is also crucial as the retrieval performance decrease with lower number of decompositions. Since decomposing an image without imposing a limit might de-stabilize the features, 4×4 resolution is taken as the smallest possible sub-image size. Hence the best number of decomposition levels is found to be 6, resulting in 19 features.

The choice of wavelet basis is not very crucial although from the experiment, the best basis was found to be Coiflet 6-tap wavelets. The binary wavelet transform however is not suitable for use with the proposed algorithm as it does not carry enough information to perform good discrimination. The optimum number of binaries to be created for each image is found to be either 49 or 99. Increasing the number beyond 99 does not appear to improve the results, while reducing it below 49 reduces the performance quite drastically. If speed is not a crucial factor, it is better to use 99 binaries as it gives the best performance, otherwise using 49 binaries offers a comparable result. Finally the proposed algorithm lends itself quite well to solving query by other low-quality image types. Experiments showed that good results are obtained in solving images of inappropriate brightness and contrast, highly compressed images, low-resolution images,

quantized images and noisy images.

Although the proposed algorithm gives good performance in solving *query by low-quality image*, it still offers plenty of room for improvement. For example the system might face a problem if the object to background ratio of the image differs significantly from the target images. In this case, a special pre-processing algorithm to detect plain background can be developed so that all images will have their default object to background ratio, hence solving the above problem. The accuracy of the QBLI algorithm can also be further improved if some sort of text annotation can be associated with it, i.e. to use both text-based and content-based image retrieval (TBIR and CBIR) as in Figure 2.1. The TBIR can act as a pre-filter for the content-based retrieval process applied at later stage.

Chapter 4

Texture Feature Extraction

In this chapter, texture representation schemes are reviewed and an evaluation of several texture features is conducted and they are compared with wavelet-based texture features. A novel implementation of the discrete wavelet frames is introduced and evaluated for texture extraction.

4.1 Introduction

Texture analysis is one of the main topics in computer vision. There is a large volume of research on different approaches and it is not possible to cover all of them in this literature review. Several popular methods and some other new approaches are chosen as a representative cross section to review. These methods belong to statistical, model-based, signal processing and structural categories. The reason why structural methods are not considered in this thesis is because they possess a number of impracticalities for our application; for example some have many pre-processing stages in order to extract texture primitives, while most of them are computationally intensive, and the assumption is that the input images contain strong regular features. There are several issues that this chapter is concerned with:

- Computational intensity. Although a lot of methods are being developed and have a high classification rate, the computation intensity needs to be considered. The goal of this thesis is to develop a texture matching algorithm for an interactive content-based retrieval application. Such a method should be as rapid as possible to avoid the user waiting for the query result. Furthermore, the number of images in the museum database for this thesis is in the region of tens of thousands of images, and could be up to hundreds of thousands in the future, which will requires a lot of time to extract features from all of the images. However this factor is not so crucial since it does not involve interaction with users.

- Properties of algorithm. It is important to identify whether a texture matching technique supports translation, rotation and scale invariance. Nearly all the approaches are translation invariant, and some of them include rotation invariance. However, there are very few studies on rotation and scale invariant texture features, since scale is a difficult task to solve in the context of texture.

- Number of textures for testing. A number of texture papers only evaluated a small subset of Brodatz textures which are treated as a standard testing set by researchers. However, evaluating only a few textures can cause a number of problems:

  - It is very difficult to find the best method based on results reported in the papers.
  - There are some textures in the Brodatz set that are known to have a low retrieval rate. We believe that a method should be tested on the entire set to see the results on the mixtures of different kinds of patterns, since in practice we should not expect an image database to always have clear and well defined textures.
  - The developer would find it difficult to choose a texture method to suit his or her own purpose. For example, a person may want to develop a technique that has the best result on lace patterns from the Brodatz set, disregarding other texture result.

- Classification vs retrieval. Most texture analysis papers evaluated their texture features based on texture classification or segmentation. Since our application is image retrieval, this may lead to inaccurate performance measures if the texture methods are evaluated based on their classification ability. In this thesis, all texture retrieval performance is evaluated in terms of precision and recall. Based on information retrieval theory, precision is defined as the fraction of images in the answer to the query that are relevant, and recall is defined as the fraction of the relevant images in the answer to the query. They are calculated as below:

$$Precision = \frac{\text{Number of retrieved images that are relevant}}{\text{Total number of retrieved images}} \tag{4.1}$$

$$Recall = \frac{\text{Number of relevant images that are retrieved}}{\text{Total number of relevant images}} \tag{4.2}$$

  Precision and recall are normally represented on a graph on apposing axis. This data is created by calculating the precision and recall up to each of the rankings in turn, so forming a graph containing as many points as documents retrieved. A perfect retrieval where only relevant documents are retrieved would form a straight line. Examples of a precision-recall graph are shown in Figure 4.1. By calculating the area under the graph, we obtain the average precision for the query. Recall is

also sometimes termed as recognition rate, and is also used individually to measure the progress of the retrieval rate against the number of retrieved documents. In this thesis, to compare the retrieval performance precisely, we will refer to the precision-recall plot and the recognition rate at several distinct places.

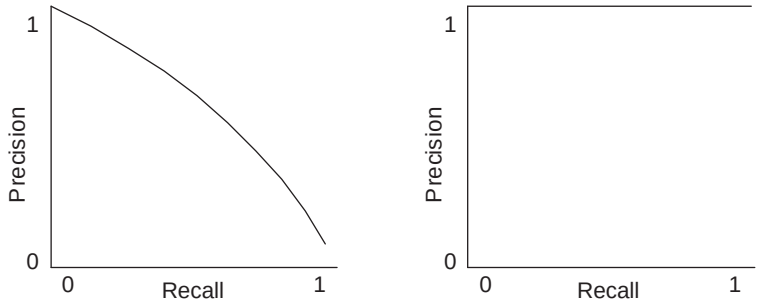


FIGURE 4.1: (left) Precision-recall graph example (right) Perfect precision-recall graph

The following section describes several popular texture feature methods, including the three different wavelet-based methods. After the review, a retrieval performance experiment is undertaken and evaluated between several selected texture features.

4.2 Texture Feature Method

In this section, all of the reviewed methods fit into either the statistical, model-based or signal processing group as they are adopted by the majority of researchers. Geometrical (structural) approaches will not be covered here for the reasons mentioned beforehand.

4.2.1 Co-occurrence Matrix

Co-occurrence matrix (or spatial grey level dependency matrix) is one of the primary techniques adopted by Haralick *et al.* (66) for texture classification. The matrix P basically contains the information of the population of two pixel grey levels with distance d at an orientation of θ. For G grey levels in the image I, P will be of size $G \times G$. The $G \times G$ grey level co-occurrence matrix P_d for a displacement vector $d = (dx, dy)$ is defined as follows. The entry (i, j) of P_d is the number of occurrences of the pair of grey levels i and j which are distance d apart. Formally, it is given as:

$$P_d(i,j) = |\{((r,s),(t,v)) : I(r,s) = i, I(t,v) = j\}| \tag{4.3}$$

where $(r,s),(t,v) \in N \times N$, $(t,v) = (r + dx, s + dy)$, and $|.|$ is the cardinality of a set.

As an example, consider the following 4×4 image containing 4 different grey values:

$$\begin{matrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \end{matrix}$$

The 4×4 grey level co-occurrence matrix for this image is given as follows:

$$P_{(0,1)} = \begin{bmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix} \qquad P_{(1,0)} = \begin{bmatrix} 6 & 0 & 2 & 0 \\ 0 & 4 & 2 & 0 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix}$$

$$P_{(1,1)} = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{bmatrix} \qquad P_{(1,-1)} = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 0 & 2 & 4 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Here the entry $(0, 0)$ of $P_{(0,1)}$ is 4 because there are four pixel pairs of $(0, 0)$ that are offset by $(0, 1)$ amount. The co-occurrence matrix reveals certain properties about the spatial distribution of the grey levels in the texture image. For example, if most of the entries in the matrix are concentrated along the diagonals, then the texture is coarse with respect to the displacement vector d. Haralick has proposed a number of useful texture features that can be computed from the co-occurrence matrix, and four of them are given below:

$$\text{Angular Second Moment} = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} p_{ij}^2 \tag{4.4}$$

$$\text{Contrast} = \sum_{n=0}^{G-1} N^2 \{ \sum_{|i-j|=n} p_{ij} \} \tag{4.5}$$

$$\text{Correlation} = \frac{1}{\sigma_x \sigma_y} \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} ij p_{ij} - \mu_x \mu_y \tag{4.6}$$

$$\text{Entropy} = \sum_{i=0}^{G-1} \sum_{j=0}^{G-1} p_{ij} log p_{ij} \tag{4.7}$$

where μ_x, μ_y, σ_x and σ_y are the means and standard deviations corresponding to the distributions

$$p_i^{(x)} = \sum_{j=0}^{G-1} p_{ij} \qquad p_i^{(y)} = \sum_{i=0}^{G-1} p_{ij} \tag{4.8}$$

Each of these statistics can be computed from each of the four co-occurrence matrices

to make up the texture features. There are also several modified version of the co-occurrence matrix, such as the generalized co-occurrence matrix by Davis et al. (106). Instead of capturing the spatial grey level difference between pixels, they defined another form of matrix to carry the distribution of local maxima. Other modified versions of the co-occurrence matrix include the work by Tamura et al. which will be discussed next.

4.2.2 Tamura's Texture Feature

Motivated by the psychological studies in human visual perception of texture, Tamura et al. explored the texture representation from a different angle (41). They developed computational approximations to the visual texture properties found to be important in psychological studies. The six visual texture properties were coarseness, contrast, directionality, line-likeness, regularity and roughness. One major distinction between the Tamura texture representation and the co-occurrence matrix representation is that all the texture properties in Tamura representation are visually meaningful, whereas some of the texture properties used in co-occurrence matrix representation may not be.

For computing coarseness, each point in the image (except the boundary) is first averaged over the neighbourhood size by using a mask operator with various sizes. The the difference between each pair of non-overlapping averaged neighbourhoods is taken on both opposite sides of the mask centre point in horizontal and vertical directions. Finally, the coarseness feature can be computed as:

$$\text{Coarseness} = \frac{1}{m \times n} \sum_{i}^{m} \sum_{j}^{n} S_{best}(i,j) \tag{4.9}$$

where m and n are the width and height of the image, $S_{best}(i,j) = 2^k$ and k is the size of the operator which gives the highest difference on either horizontal or vertical directions.

Contrast was evaluated on the basis of standard deviation, polarization of grey level range, sharpness of edges and period of repeating patterns. Tamura et al. used the kurtosis method to measure the polarization with the ratio of fourth moment and square of variance. Then the contrast is measured with kurtosis, α and standard deviation σ as the following:

$$\text{Contrast} = \frac{\sigma}{\alpha_4^n} \tag{4.10}$$

where n was evaluated with various values, and $n = \frac{1}{4}$ yielded the best correlation result to human visual inspection on contrast.

The assessment of directionality was applied with horizontal and vertical edge operators. The magnitude ΔG and the edge direction θ of each centre pixel of an operator can be extracted with $|\Delta G| = (|\Delta_H| + |\Delta_V|)/2$ and $\theta = tan^{-1}(\frac{\Delta_V}{\Delta_H}) + \frac{\pi}{2}$, where Δ_V and Δ_H were the values after the convolution of vertical and horizontal edge operators respectively. A

directionality histogram is then obtained by counting the points where the magnitude is greater than or equal to 12 and its edge direction is in a certain quantized degree range. In order to measure the directionality over the histogram, the directionality feature is derived from the sharpness in peaks in which the second moment of each peak between valleys was calculated.

$$\text{Directionality} = 1 - r \cdot n_p \cdot \sum_{p}^{n_p} \sum_{\phi \in w_p} (\phi - \phi_p)^2 \cdot H_D(\phi) \tag{4.11}$$

where n_p is the number of peaks, ϕ_p is the p-th peak position of directional histogram H_D, w_p is the range of p-th peak between valleys, r is the normalizing factor for ϕ, and ϕ is the quantized direction code from 0 to 15.

Line-likeness is computed by first constructing a co-occurrence matrix P_D, with quantized direction codes and magnitude which are used for the directionality feature. An element in the matrix means the number of co-occurrences that two points p and q both have quantized direction i and j respectively and magnitude greater or equal to 12 with d_4 distance apart.Then the line-likeness feature is calculated as:

$$\text{Line-likeness} = \sum_{i}^{n} \sum_{j}^{n} P_D(i,j) \frac{cos[(i-j)\frac{2\pi}{n}]}{\sum_i^n \sum_j^n P_D(i,j)} \tag{4.12}$$

For calculating regularity, Tamura et al. assumed that it was based on all the above features varied over the image. The smaller variation of each feature gave a higher regularity. Each feature wac evaluated with a sub-image from the original image and the standard deviation was calculated among these sub-images by a normalizing factor and subtracted from 1.

$$\text{Regularity} = 1 - r(\sigma_{coarseness} + \sigma_{contrast} + \sigma_{directionality} + \sigma_{line-likeness}) \tag{4.13}$$

where r is the normalizing constant, $\sigma_{xxx}$is the standard deviation of each above feature taken from 9 sub-images.

Finally, in order to compute roughness, Tamura et al. suggested that it was derived from the level of coarseness and brightness.

$$\text{Roughness} = \text{Coarseness} + \text{Contrast} \tag{4.14}$$

The above six features of the Tamura texture representation have the advantage in that they can provide a more user-friendly interface, and have been proved very useful in texture representation.

4.2.3 Simultaneous Autoregressive Model (SAR)

SAR is a well known statistical model for texture classification, segmentation and synthesis. The simultaneous autoregressive model for an image $I(m,n)$ can be expressed as:

$$\begin{aligned} I(m,n) &= \textstyle\sum_{(k,l)\in N} \theta(k,l) I(m-k,n-l) + \sigma_\varepsilon \varepsilon(m,n), \\ \\ \textit{where} \quad & N\text{is the model neighbourhood,} \\ & \theta(k,l)\text{are the model parameters,} \\ & \sigma_\varepsilon \varepsilon(m,n)\text{the model error terms} \end{aligned} \tag{4.15}$$

Equation 4.15 generally describes the centre pixel which has a spatial relation $\theta(n)$ with neighbouring pixels. The neighbouring pixels can be first, second order etc. For a second order neighbourhood, four parameters θ_1, θ_2, θ_3 and θ_4 represent the four different directions (0°, 90°, 45°, 135°) of the model. A series of equations with unknown parameters θ and σ_ε are generated by applying the SAR formula to each pixel of the image. Two conventional techniques are used to estimate the unknown parameters, the Maximum Likelihood (ML) and Least Square (LS) methods. The least square method is more generally accepted because it is easier and faster although the results are inconsistent (69).

Mao et al. (70) showed that increasing the number of SAR parameters (larger neighbourhood on a single resolution) reduced the discrimination power as it has a severe averaging effect on some of the good distinguishable parameters. They demonstrated that small neighbourhoods a few pixels apart can provide satisfactory discrimination on lower resolutions. In order to capture the details of different resolutions, they used the Gaussian Image Pyramid model to construct a low pass filter and sub-sampled the image. The size of output images is reduced through different gaussian image levels from fine to coarse. Each of these output images was then integrated with the SAR technique to provide the parameters θ and σ_ε on each level. This modified version is known as the multiresolution simultaneous autoregressive model.

4.2.4 Markov Random Field

Cross and Jain (107) introduced the use of Markov Random Filed (MRF) properties to generate texture-based features and synthesizing textures with given parameters. Since then MRF has been widely studied for texture modelling. The MRF assumes that the state of an image pixel is highly dependent on the brightness values of neighbouring pixels (conditional probability) and also the configuration of the neighbouring pixels. In MRF, different orders can be used and higher order has a larger number of neighbour pixels and a different coding scheme. A second order is used in normal cases with four codings over the image lattice.

Cross and Jain used the binomial model to model a pixel grey level value with a binomial distribution by taking the parameters of neighbourhood brightness values. These neighbouring parameters are evaluated according to the order of neighbourhoods, i.e. the second order is 8 neighbourhoods. The number of tries in the binomial model is the number of grey levels that appeared in the image taken for analysis. The main advantage of the MRF is that the parameters can capture the properties of a random texture very well, and can generate a visually similar texture for inspection. In (68), Chellappa and Chatterjee further improved the MRF by using a Gaussian Markov Random Field model to represent textures.

4.2.5 Fractal Dimension

Fractals were pioneered by Mandelbrot (108). They have been widely used in computer graphics and other problems of numerical analysis. In texture analysis, Pentland (71) investigated the property of the fractal dimension (FD) as relating to natural textures. The author concluded that fractals provide good model for describing the roughness of textured surfaces. Intuitively, the larger the fractal dimension, the rougher the texture is. The FD also shows correspondence to human perception, and is possibly suited to natural or satellite image analysis, with changing FD values.

Keller et al. (109) propose a method for estimating the fractal dimension. The image is first represented by points in 3 dimensions as $(x, y, f(x, y))$. A cube with size $s \times s \times s$ is then overlayed onto each image pixel and the number of points that are in the cube are counted. This procedure is repeated for different sizes of cube, L, on every pixel. The fractal dimension can be approximated by using the least square method with different size cubes. However, some textures may have different appearances but with the same fractal dimension. Keller et al. suggested an extra parameter, *lacunarity*, which has been defined by Mandelbrot to distinguish the same FD with different appearances.

Sarkar and Chauduri (110) propose a more effective and noise tolerant method known as Differential Box Counting (DBC), to estimate fractal dimensions. Six features are derived, each describing the fractal dimension of different properties of an image, such as the FD of the original image, the FD of the high/low gray-valued image, the FD of the horizontally/vertically smoothed image and the multi-FD of an image.

4.2.6 Law's Texture Feature

The texture energy measures developed by Laws (111) at the University of Southern California have been used for many diverse applications. These measures are computed by first applying small convolution kernels to a digital image, and then performing a nonlinear windowing operation. The 2-D convolution kernels typically used for texture discrimination are generated from the following set of one-dimensional convolution

kernels of length five:

$$\begin{array}{rcl} L5 &=& [\ \ 1 \ \ \ 4 \ \ 6 \ \ \ 4 \ \ \ 1\ ] \\ E5 &=& [\ -1 \ -2 \ \ 0 \ \ \ 2 \ \ \ 1\ ] \\ S5 &=& [\ -1 \ \ \ 0 \ \ 2 \ \ \ 0 \ -1\ ] \\ W5 &=& [\ -1 \ \ \ 2 \ \ 0 \ -2 \ \ \ 1\ ] \\ R5 &=& [\ \ 1 \ -4 \ \ 6 \ -4 \ \ \ 1\ ] \end{array} \tag{4.16}$$

These mnemonics stand for Level, Edge, Spot, Wave, and Ripple. Note that all kernels except L5 are zero-sum. In his dissertation, Laws also presents convolution kernels of length three and seven, and discusses the relationship between different sets of kernels.

From these one-dimensional convolution kernels, we can generate 25 different two-dimensional convolution kernels by convolving a vertical 1-D kernel with a horizontal 1-D kernel. As an example, the L5E5 kernel is found by convolving a vertical L5 kernel with a horizontal E5 kernel. Of the 25 two-dimensional convolution kernels that we can generate from the one-dimensional kernels above, 24 of them are zero-sum; the L5L5 kernel is not.

Given an image, we first apply each of our 25 convolution kernels to the image. This results in a set of 25 grey scale images. The energy of each grey scale image is computed as texture features. The energy calculation is given by:

$$e = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} |W(i,j)| \tag{4.17}$$

4.2.7 Discrete Cosine Transform (DCT)

The discrete cosine transform is popular in image coding due to its good performance and fast implementation (112). It is, for instance, the backbone in the JPEG compression standard. Ng et al. (113) suggest using a 3×3 discrete cosine transform for texture feature extraction.

Image transforms are equivalent to critically sampled filter banks. The 3×3 discrete cosine transform is tested in a filter bank implementation, but without critical sampling. The filter bank is separable, determined by the one-dimensional filter masks:

$$\begin{array}{rcl} h_1 &=& [\ 1 \ \ \ 1 \ \ \ 1\ ] \\ h_2 &=& [\ 1 \ \ \ 0 \ -1\ ] \\ h_3 &=& [\ 1 \ -2 \ \ \ 1\ ] \end{array} \tag{4.18}$$

Note that all kernels except h_1 are zero-sum. The result of the discrete cosine transform is 9 grey scale images, each of which is computed for its energy using the formula in

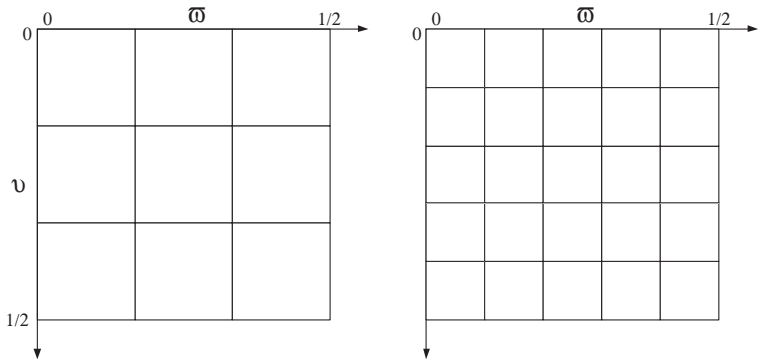


FIGURE 4.2: Frequency coverage of (left) DCT features, and (right) Law's features

equation 4.17. Ng et al. further suggest that the low-frequency component of the DCT is excluded, thus yielding only eight features. Figure 4.2 shows the difference in frequency coverage between DCT features and Law's texture features.

4.2.8 Gabor Transform

Gabor functions (114) are Gaussians modulated by a complex sinusoid. A well known property of these functions is that they achieve the minimum possible joint resolution in the space and frequency domain. A two-dimensional Gabor function takes the form of:

$$g(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} exp\left[-\frac{1}{2}\left(\frac{x^2}{{\sigma_x}^2} + \frac{y^2}{{\sigma_y}^2}\right) + 2\pi jWx\right] \tag{4.19}$$

where σ_y and σ_x are the standard deviations of the Gaussian envelope along the x and y directions respectively, and W is the frequency of the sinusoidal function.

Gabor functions are also known as Gabor wavelets because there is a corresponding wavelet (B-spline) from which it differs only by a small correction term. Let $g(x,y)$ be the mother Gabor wavelet, S be the total number of scales, and K be the total number of orientations (or translations) to be computed. A family of Gabor wavelets can then be obtained by appropriate dilations and rotations of $g(x,y)$ through the generating function:

$$\begin{aligned} g_{mn}(x,y) &= a^{-m} g(x',y') \\ x' &= a^{-m}\left(xcos\theta + ysin\theta\right) \\ y' &= a^{-m}\left(-xsin\theta + ycos\theta\right) \end{aligned} \tag{4.20}$$

where $a > 1$, $\theta = n\pi/K$, and m, n are integers, $m = 1, 2, ..., S$ and $n = 1, 2, ..., K$. Expanding the mother Gabor wavelet forms a complete but non-orthogonal basis set. This non-orthogonality implies that there will be redundant information between different resolutions in the output data. To reduce this redundancy, Manjunath and Ma (114) proposed the following strategy: Let U_l and U_h denote the lower and upper frequency of interest. Then the design strategy is to ensure that the half-peak magnitude support

of the filter responses in the frequency spectrum touch each other as shown in Figure 4.3, for $S = 4$ and $K = 6$.

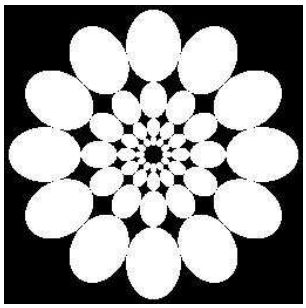


FIGURE 4.3: Frequency spectrum view of 2D Gabor transform.

The Gabor transform is then defined by:

$$W_{mn}(x, y) = \int I(x_1, y_1) g_{mn} * (x - x_1, y - y_1) dx_1 dy_1 \tag{4.21}$$

where $*$ indicates the complex conjugate. The Gabor transform therefore produce KxS number of output images, and features can be computed from each of the output images using the formula in equation 4.17.

4.2.9 Wavelet-based Texture Features

As mentioned in the previous chapter, there are two types of wavelet transform, the continuous wavelet transform and the discrete wavelet transform. The continuous wavelet transform offers more variety in frequency tuning or orientation, while the discrete wavelet transform only offers four orientations per scale, but has the important advantage of fast implementation. In this thesis we are only interested in the discrete wavelet transform because of its low computational intensity. Other wavelet-based feature extraction techniques such as the nonseparable wavelet transform (115), the complex wavelet transform (116) and rotated wavelet filters (117) require quite a high computational intensity because they are based on the continuous wavelet transform, thus possess no advantage over other techniques in terms of speed. In fact the Gabor transform discussed in the previous section can be used to represent the family of continuous wavelet transform techniques for performance comparison purpose.

The three most popular methods among the discrete wavelet transforms are the pyramidal wavelet transform (PWT) (93), tree-structured wavelet transform (TWT) (77; 118) and discrete wavelet frames (DWF) (78), see Figure 2.8. PWT decomposes an image into different frequency and orientation channels by repeatedly decomposing the low frequency channels of the image. This results in $3L + 1$ channels, where L is the depth of the decomposition. DWF, or over-complete wavelet transform, is nearly identical to

the PWT, except that one up-samples the filters, rather than down-samples the image. While the frame representation is over-complete, and computationally more intensive than PWT, it holds the advantage of being translationally invariant (78). Given an image, the DWF decomposes it using the same method as the PWT, but without the sub-sampling process. This results in four channels with the same size as the input image. The decomposition is then continued in the LL channels only as in PWT, but since the image is not sub-sampled, the filter has to be up-sampled by inserting zeros in between its coefficients.

Unlike PWT, the TWT repeatedly performs the decomposition not only in the low frequency channels, but in the medium and high frequency channels as well. TWT however has several varieties of decomposition strategy. Laine and Fan (118) perform a full decomposition tree, while Chang and Kuo (77) proposed an adaptive decomposition, where only channels with energy above certain threshold are decomposed further. The full decomposition tree results in a large number of features while the adaptive decomposition tree results in complexity in indexing. For most applications, only the low and the medium channels are decomposed further, resulting in medium size features. The high frequency channels are not decomposed since it destabilizes the texture features (91). In this thesis we will also be using the suggested tree structure for our experiments. This will results in $4(3^L + 3^{L-1} + ... + 3^0)$ channels.

For all three methods, features are extracted by computing the mean energy from each channel (equation 4.17), just as other signal processing based texture features.

4.3 Experimental Evaluation

Four major experiments are conducted. The first three experiments were based on a grey scale database, and are used to identify the best starting point for the development of an image retrieval application, configuring the parameters of the chosen method to achieve the optimum performance, as well as improving it using some statistical analysis. The last experiments were involved in testing the configured method on a colour texture database. The database used in the retrieval experiments for the grey scale image is the Brodatz texture database, while the Vision Texture database is used for the colour image database.

Although experiments matching sub-images are very standard among most of the texture papers, it only illustrates to the readers how accurate a technique is on sub-images. Also the Brodatz textures album was not designed for image recognition purposes, it is questionable whether this collection of textures can cover a wide range of possible textures. Moreover a texture technique may be able to match all the correct sub-images in the top matches but it may not perform well on visual similarity after those accurate matches. One should experiment on the entire Brodatz textures database and other

texture images with human visual assessment on a large set of human subjects. This can assist the researchers and readers to another dimension in performance evaluation, although such examination can be very expensive in cost and time. Nevertheless, the evaluation of sub-image matching is still common in the field of texture analysis, as this quickly gives a first impression of the performance and of course this depends on the size of the database.

The Brodatz database contains 112 textures of various kinds, including the many inhomogeneous ones which are not usually included in texture studies. By including the entire Brodatz collection in the database, we allow the potential of confusion and failure that exists when texture algorithms encounter non-texture regions in natural scenes. Each Brodatz texture is scanned at 300 dpi with 256 grey levels, and is 512×512 in resolution. Each 512×512 image was then cut into 16 non-overlapping sub-images of size 128×128. A total of 1792 (112×16) database images are produced from the texture album. The image classes are defined by the 112 Brodatz textures.

Vision Texture is a collection of texture imagery developed at MIT that is publicly available for evaluating different texture features. However, hardly any studies use this texture set for evaluating texture analysis performance. It contains 4 components: reference textures, texture scenes, video textures and video orbits which all contain a set of different textures for various experimental purposes. In this report, the reference textures are used for evaluating the discrete wavelet frames technique on colour textures. There are 167 colour textures in the reference textures components. Like in the experiment with the Brodatz textures, all images are cut into 16 non-overlapping 128×128 sub-images, yielding a total 2672 database images in the VisTex collection.

For each experiment, each image in the database is used once as a retrieval prototype and the average precision and recall rate is computed for different numbers of retrieved images. A 100% recall rate is achieved at anytime all 15 matches are found within the top retrieved images considered, R, while a 100% precision rate is achieved only when all 15 matches are found within the top 15 matches. For example, if 9 matches were found in the first 15 retrieved images, while 6 others were found in the top 30, then the recall rate is $\frac{9}{15} \times 100 = 60\%$ at $R = 15$, and reaches 100 % at $R = 30$. The precision rate is computed as $\frac{9}{15} \times 100 = 60\%$ and $\frac{9}{30} \times 100 = 30\%$ respectively. As mentioned before, each experiments will be compared using a precision-recall plots and an average recognition rate. The average recognition rate will be measured at $R = 15$,30 and 45, and is presented in a table to support the precision-recall plot measurement.

4.3.1 Evaluation of Texture Features

In this section, all three wavelet-based texture feature extraction techniques along with several other techniques will be evaluated for their performance in retrieving similar

texture from the Brodatz texture database. For performance comparison purposes, at least one representative from the statistical, model-based and signal processing categories will be evaluated. The discrete cosine transform (DCT), the Law's texture feature and the Gabor transform techniques represent the signal processing categories (the Gabor transform can also represents the continuous wavelet transform family), while the grey level co-occurrence matrix (GLCM) and the multiresolution simultaneous autoregressive model (MRSAR) represents the statistical and model based categories respectively. The parameters used for each methods are summarized below:

- PWT: A three level wavelet decomposition is used using the Daubechies 8-tap wavelet basis, resulting in 10 features.
- TWT: Same as PWT, but all channels except the *HH* channels are decomposed further, resulting in 40 features.
- DWF: Same as PWT, 10 features.
- Gabor: The method described in the previous section which helps in reducing redundant information is used. Two different parameters are tested, the first one having 6 orientations and 4 scales (24 features), while the second having 6 orientations and 3 scales (18 features).
- Law's: As described in the previous section, 25 features.
- DCT: As described in the previous section, with the low frequency channels excluded, 8 features.
- GLCM: To get the best statistical significance, the 255 grey level image is first quantized to only 8 grey levels as suggested by Ohanian and Dubes (90), and the angular second moment, contrast, correlation and entropy are computed from four different orientations ((0,1),(1,0),(1,1) and (1,-1)) as described in the previous section, yielding 16 features.
- MRSAR: Features are computed from three neighbourhoods as shown in Figure 4.4 with 4 parameters and the standard deviation of the error term from each neighbourhoods are used as features, yielding 15 features.

We tested two different parameters for the Gabor transform in order to observe a fairer comparison with the wavelet-based techniques. In several texture techniques comparison papers, when evaluating the performance between wavelet-based techniques and the Gabor transform, we believe a rather unfair comparison was made. The Gabor transform used for the comparison are usually with six orientations and four scales. This is compared with the wavelet-based techniques using 3 decomposition levels, i.e. three scales. We believe in order to obtain a fairer comparison the same number of scales should be used. Although one might argue that the Gabor transform with six orientations and

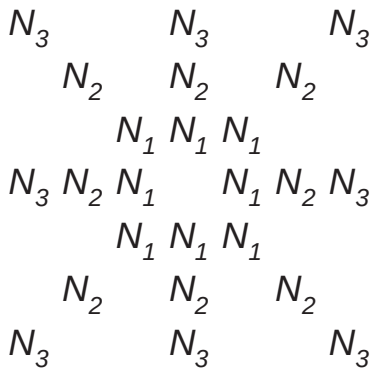


FIGURE 4.4: Neighbourhood sets N_1, N_2 and N_3 for the MRSAR features. Each N_i corresponds to one relative pixel position

four scales are the best representatives of the Gabor transform, and therefore should be used for comparison purposes, it is interesting to also observe how the two techniques fare for the same number of scales. Therefore two different scales are used for the Gabor transform features.

To compute the features of the signal processing-based techniques (PWT, TWT, DWF, Gabor, DCT, Law's), the mean absolute value or mean energy of each channel or filtered image is used and is given as:

$$f = \frac{1}{M \times N} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} |W_k(i,j)| \tag{4.22}$$

where M and N are the number of rows and columns of the channels or filtered images, and W_k is the k-th channels or filtered images.

The distance measure used for comparing and retrieving the similar patterns is defined to be:

$$d(i,j) = \sum_k \left| \frac{f^{(i)}(k) - f^{(j)}(k)}{\sigma(k)} \right| \tag{4.23}$$

where i and j denote two image patterns, $f(k)$ is the k-th component features, and $\sigma(k)$ is the standard deviation of the distribution of features $f(k)$ in the entire database and is used to normalize the individual feature components.

4.3.1.1 Accuracy

Figure 4.5 shows the precision-recall plot for the nine methods, while Table 4.1 gives the recognition rate. In the graph, the point where the slope of the graphs change is when $R = 15$, and is more or less similar to the data given in the corresponding table for $R = 15$. It is therefore easier to refer to this point ($R = 15$) when comparing

recognition rate for a particular method. From the figures, the best texture matching method is found to be the MRSAR technique which is just slightly better than the Gabor transform method of 6 orientations and 4 scales. The performance of the three wavelet-based method are almost identical with the DWF favored just slightly from the other two. The TWT does not seem to give a much better retrieval rate than the PWT, instead it is slightly worse than the PWT and the DWF. This might be because the decomposition on the medium frequency channels (LH and HL), although it may provide better discrimination features for some textures, it might also lead to poor discrimination for others. The fact that DWF carries more information than PWT and TWT in terms of coefficient is probably the reason it is better than the other two.

However all three wavelet-based techniques are found to be better than the Gabor transform with 6 orientations and 3 scales. This means that for 3 scales decomposition, the discrete wavelet transform technique is better than the continuous wavelet transform technique, even though the Gabor transform offers more orientation tunings (6 vs 4). However all three wavelet-based techniques are inferior to the Gabor transform with 6 orientations and 4 scales. The Law's texture feature and the discrete cosine transform give a considerable performance but not as good as the MRSAR, Gabor and wavelet-based techniques. Note that the Law's texture feature is considerably better than the discrete cosine transform since it partitions the frequency space into 25 regions instead of just 9 for the DCT, and thus has better discrimination features. The grey level co-occurrence matrix gave the worst performance among the nine evaluated techniques. Appendix A gives the detailed recognition rate for all 112 Brodatz textures for all 9 methods evaluated.

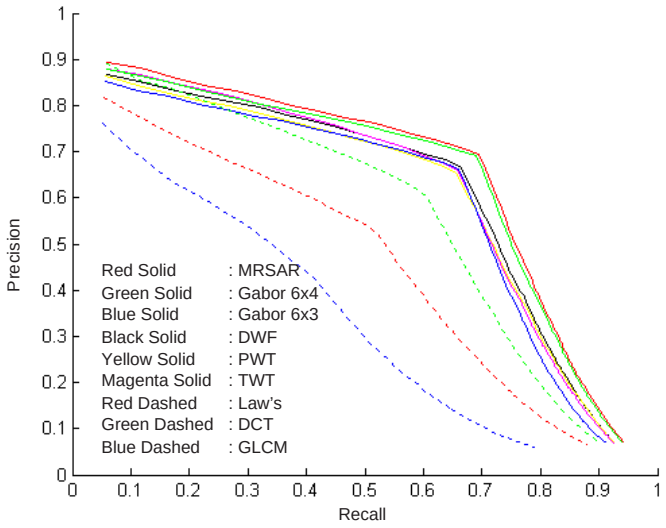


FIGURE 4.5: Precision-recall plot for nine texture methods

From Appendix A, some texture classes with a very low recognition rate are observed. For example, texture D043 gave a very low average recognition rate of less than 10%

Texture Method	Number of retrieved images considered		
	15	30	45
PWT	65.47	76.72	81.36
TWT	65.87	76.29	80.90
DWF	66.42	77.42	81.61
Gabor 6×3	66.12	75.47	79.53
Gabor 6×4	69.06	78.98	83.41
DCT	52.20	64.28	70.56
Law's	60.44	71.53	76.55
GLCM	41.86	52.68	58.92
MRSAR	69.40	79.36	83.85

TABLE 4.1: Percentage of recognition rate for different texture methods

of its entire class member. After visual inspection, texture D043 is a nonhomogeneous texture, therefore when one of the sub-images of the texture is used as the query, the retrieved result will mostly contain textures from other classes. Figure 4.6 show the 16 sub-images of texture D043 and D044. Clearly if one of the sub-images are taken as query for texture retrieval, it could easily be confused between the two classes. Furthermore, some of the sub-images do not even contain any texture, which will definitely result in incorrect retrieval. To obtain a much better accuracy for our evaluation, we will only include the homogeneous texture classes as the query prototype. The nonhomogeneous texture will remain in the database to provide the non-texture regions, but will not be used as query. Thus the image database remains with size 1792 images. After careful observation, 12 texture classes (D013, D042, D043, D044, D045, D058, D059, D061, D069, D090, D091 and D097) will be excluded from the experiments. All 12 textures are shown in figure 4.7. The evaluation will now be based on only 100 homogeneous texture classes, and will reflect a more *true* accuracy of the retrieval performance.

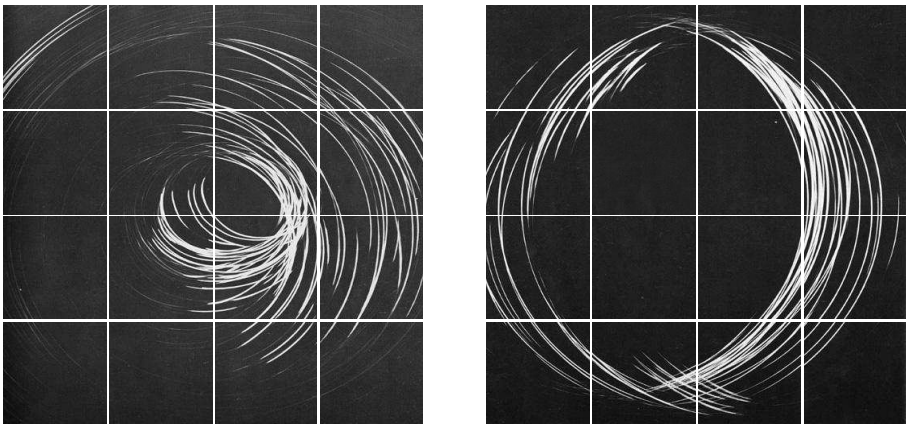


FIGURE 4.6: The 16 sub-images of texture (left) D043 and, (right) D044

Figure 4.8 and table 4.2 shows the performance for just the 100 homogeneous Brodatz textures. From the table and graph, all nine texture methods experienced a significant rise of recognition rate compared to using the whole Brodatz texture sets. In terms of

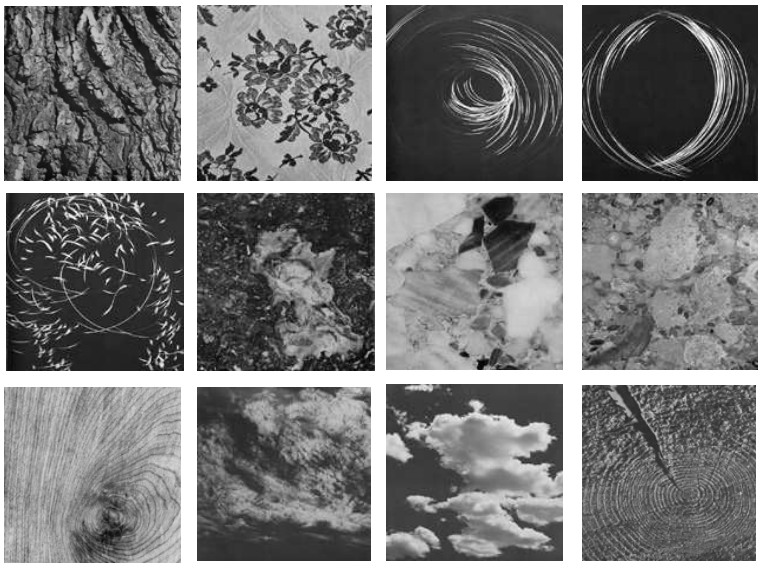


FIGURE 4.7: 12 nonhomogeneous textures excluded from the evaluation

evaluation of the best method, the rankings between texture method does not change much, except for the 6 orientations and 3 scales Gabor transform which is now closer to the performance of the wavelet-based methods. The best recognition rate for top 15 retrieved images was recorded by the Gabor transform with 6 orientations and 4 scales and the MRSAR method with around 74% accuracy with the wavelet-based method not far behind at 71% accuracy. All the following experiments on grey scale textures will therefore be based on these 100 homogeneous textures only, as they reflects a more *true* performance.

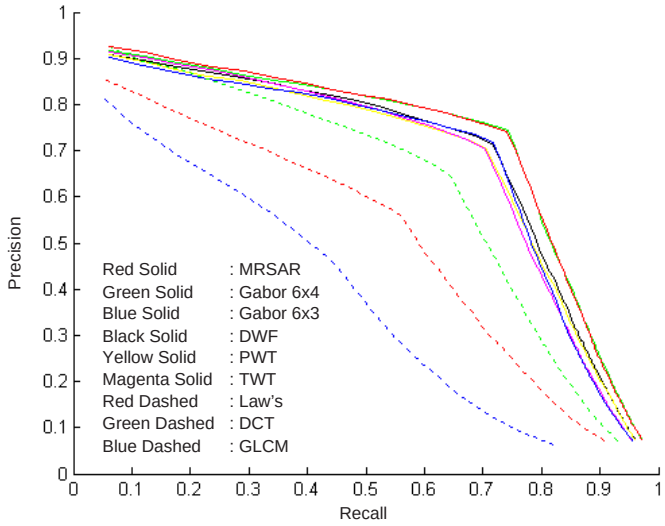


FIGURE 4.8: Precision-recall plot for nine texture methods using only 100 homogeneous textures

Texture Method	Number of retrieved images considered		
	15	30	45
PWT	70.48	82.07	86.55
TWT	70.42	81.02	85.52
DWF	71.56	82.60	86.84
Gabor 6×3	71.77	81.41	85.38
Gabor 6×4	74.34	84.46	88.63
DCT	55.96	68.53	74.95
Law's	64.49	75.98	80.92
GLCM	44.99	56.18	62.52
MRSAR	74.10	84.30	88.48

TABLE 4.2: Percentage of recognition rate for different texture methods using only 100 homogeneous textures

Texture Method	128×128	256×256
PWT	0.09	0.21
TWT	0.45	0.95
DWF	0.19	0.58
Gabor 6×3	2.23	53.30
Gabor 6×4	2.78	59.11
DCT	0.06	0.57
Law's	0.17	2.23
GLCM	1.27	3.68
MRSAR	15.01	61.16

TABLE 4.3: Time taken to extract features in seconds for different texture methods

4.3.1.2 Speed of Computation

Table 4.3 gives the time taken to perform feature extraction on images of size 128×128 and 256×256 for all nine texture methods. The time was recorded using Matlab 6.5 on a Pentium III 733MHz processor. The PWT, TWT, DWF, DCT and Law's texture features are very fast in extracting features of size 128×128, although Law's texture method was quite affected by the increase in image sizes. This is illustrated by the fact that the time taken for Law's method was increased by a factor of more than 10 from image size 128×128 to the size 256×256.The Gabor transform and MRSAR methods is the most computationally intensive. The Gabor transform method is still considerable for image size 128×128 but it took almost a minute to compute features from image size 256×256. The performance of the GLCM is also good although not as good as the wavelet-based algorithm and the DCT and Law's techniques.

4.3.1.3 Choosing the Best Texture Method

Taking into account both the retrieval accuracy and the speed of the algorithm, we can conclude that the wavelet-based methods give the best overall performance in retrieving

texture. All the three wavelet-based texture methods give a very similar retrieval accuracy and computational speed. However, since the TWT used more features compared to the other two (40 compared to 10), we discard the TWT from consideration in choosing the best available texture features. Thus we only need to consider between PWT and DWF methods to be used for the rest of this thesis. However, since the DWF offers translation invariance (78), this might offer more advantage for the retrieval purpose.

Consider the image in Figure 4.9. The feature vectors computed for the image on the left and its shifted version on the right using DWF will produce exactly the same feature vectors. However this is not true for the PWT, which recorded different feature vectors between the two images. Although the translation invariant properties do not show any advantage in the sub-image matching experiment, we believe this property might be useful when retrieving texture from real scene images. For this reason, and also one other reason from a segmentation point of view, which will be discussed in chapter 6, we opt to use the discrete wavelet frames method as the starting point.

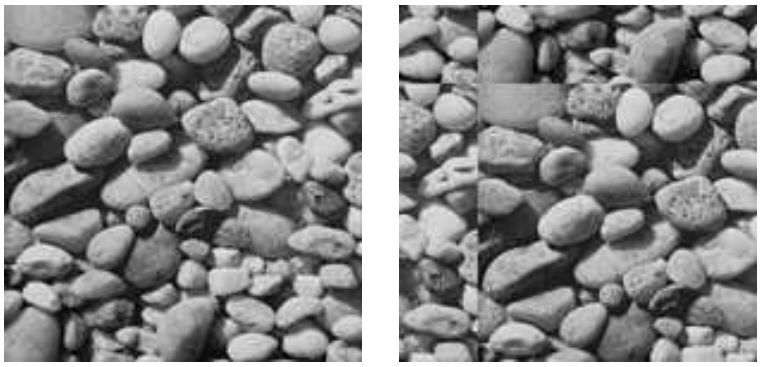


FIGURE 4.9: (left) A texture and, (right) its shifted version

4.3.2 Evaluating the Best Parameters for the DWF

In the last section, the discrete wavelet frames has been chosen as the best available texture method to be used in this thesis. In this section, a range of wavelet parameters will be tested for the DWF and their effect on the retrieval accuracy will be evaluated. The parameters to be tested include the type of the wavelet basis, the number of decomposition levels, and the type of padding used during the computation of wavelet transform. In the last section, we use a Daubechies 8-tap wavelet filter as the wavelet basis for the DWF, along with a 3 levels of decomposition using a periodic image padding.

4.3.2.1 The Choice of Wavelet Basis

There are a lot of wavelet bases that can be used for computing the wavelet transform and the wavelet frames, making it impossible to evaluate all of them within this thesis.

However, a selection of well known bases such as the Daubechies, Haar, Coiflet and Symlet wavelet will be tested for this experiment. Three different vanishing moments are chosen for each of the Coiflet and Symlet wavelet family, while two and one from the Daubechies and the Haar wavelet family respectively. Table 4.4 gives the filter coefficients for the ten wavelet bases.

Haar	Daubechies		Coiflet			Symlet		
	8-tap	16-tap	6-tap	12-tap	18-tap	4-tap	8-tap	16-tap
-0.7071	-0.2304	-0.0544	0.0727	-0.0164	0.0038	-0.4830	-0.0322	-0.0019
0.7071	0.7148	0.3129	0.3379	-0.0415	0.0078	0.8365	-0.0126	-0.0003
	-0.6309	-0.6756	-0.8526	0.0674	-0.0235	-0.2241	0.0992	0.0150
	-0.0280	0.5854	0.3849	0.3861	-0.0658	-0.1294	0.2979	0.0038
	0.1870	0.0158	0.0727	-0.8127	0.0611		-0.8037	-0.0491
	0.0308	-0.2840	-0.0157	0.4170	0.4052		0.4976	-0.0272
	-0.0329	-0.0005		0.0765	-0.7938		0.0296	0.0519
	-0.0106	0.1287		-0.0594	0.4285		-0.0758	0.3644
		0.0174		-0.0237	0.0718			-0.7772
		-0.0441		0.0056	-0.0823			0.4814
		-0.0140		0.0018	-0.0346			0.0613
		0.0087		-0.0007	0.0159			-0.1433
		0.0049			0.0090			-0.0076
		-0.0004			-0.0026			0.0317
		-0.0007			-0.0011			0.0005
		-0.0001			0.0005			-0.0034
					0.0001			
					-0.0000			

TABLE 4.4: Filter coefficients of different wavelet basis

Figure 4.10 and table 4.5 shows the average recognition rate for the ten wavelet bases in retrieving 100 homogeneous Brodatz texture classes. From the figures, it can be observed that the choice of wavelet basis does not effect the performance of the discrete wavelet frames in texture retrieval significantly, and therefore is not critical. Except from the Haar wavelet, which is the simplest of wavelet bases, all other wavelet bases give a very similar average recognition rate of about 71%. In terms of speed, there is also not a significance difference between all ten wavelet bases.

4.3.2.2 Number of Decomposition Levels

5 different decomposition levels of 1,2,3,4 and 5 will be evaluated. This results in 4,7,10,13 and 16 features respectively for the discrete wavelet frames decomposition. Figure 4.11 and table 4.6 shows the average recognition rate of the five levels. From the figures, we can conclude that increasing the number of decomposition levels helps in improving the retrieval accuracy of the discrete wavelet frames. The increment in accuracy is quite dramatic from 1 decomposition level to 3 decomposition levels, but only a small increment is observed when the decomposition levels change from 3 levels

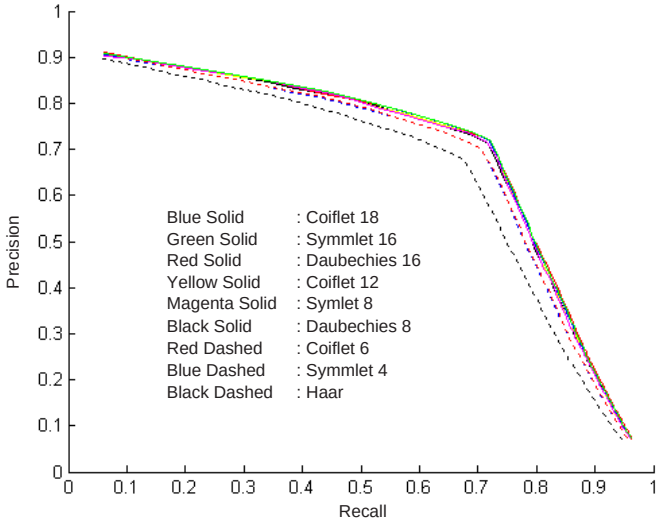


FIGURE 4.10: Precision-recall plot for different wavelet basis

Wavelet Basis	Number of retrieved images considered		
	15	30	45
Haar	67.57	79.21	83.90
Daubechies 8-tap	71.56	82.60	86.84
Daubechies 16-tap	72.02	83.11	87.39
Coiflet 6-tap	70.44	81.54	85.94
Coiflet 12-tap	71.59	82.65	86.90
Coiflet 18-tap	71.92	82.87	87.13
Symlet 4-tap	70.33	81.45	85.86
Symlet 8-tap	71.57	82.59	86.82
Symlet 16-tap	72.04	82.90	87.26

TABLE 4.5: Percentage of recognition rate for different wavelet basis

to 4 and almost no increment at all from 4 to 5 levels. This might suggest that 3 levels is the optimal choice for discrete wavelet frames decomposition of a 128×128 images. Furthermore the time taken to compute the features also increase quite dramatically when the decomposition levels increase from three to higher levels. Table 4.7 shows the time needed to compute the discrete wavelet frames features for image of various sizes. From the table, it is clear that by increasing the decomposition levels, the computational load increases almost quadratically. Since the improvement in accuracy over the 3-levels decomposition is not very significant, 4- and 5-levels decomposition can be considered as unnecessary. The optimal choice of the decomposition levels for our research purposes is therefore the 3-levels, with 10 features.

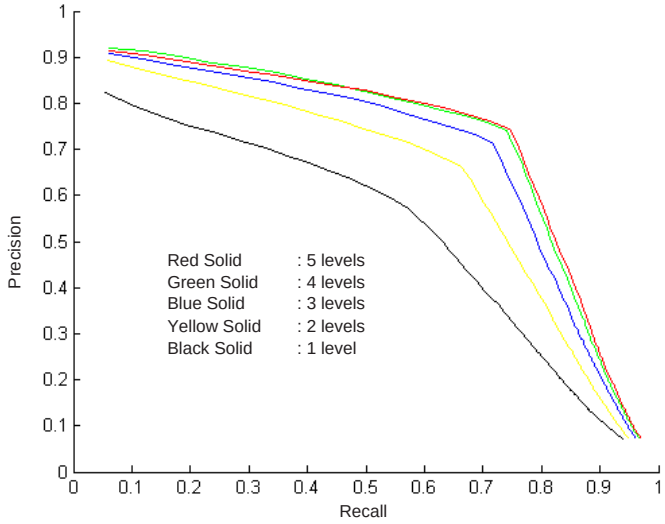


FIGURE 4.11: Precision-recall plot for different decomposition levels

Decomposition Levels	Number of retrieved images considered		
	15	30	45
1	57.29	72.60	79.14
2	66.35	79.18	84.24
3	71.56	82.60	86.84
4	74.08	84.39	88.28
5	74.56	85.07	88.85

TABLE 4.6: Percentage of recognition rate for different decomposition levels

4.3.2.3 Image Padding Type

Wavelet transform and wavelet frames is the process of filtering the image with appropriate filters. Therefore depending on the length of the filter coefficients, the image to be filtered needs to be padded before the filtering process takes place. In this section we will consider three types of image padding:

- Zero Padding: This method assumes that the signal is zero outside the original support.
- Periodic Padding: This method assumes that signals or images can be recovered outside their original support by periodic extension.
- Symmetrical Padding: This method assumes that signals or images can be recovered outside their original support by symmetric boundary value replication.

Figure 4.12 and table 4.8 show the average recognition rate for the three different image padding. It is very clear from this experiment that the choice of image padding does not

Decomposition levels	128×128	256×256	512×512
1 (4 features)	0.06	0.17	0.57
2 (7 features)	0.11	0.34	1.19
3 (10 features)	0.19	0.58	2.15
4 (13 features)	0.40	1.14	3.93
5 (16 features)	2.30	5.21	13.38

TABLE 4.7: Time taken to extract features in seconds for different decomposition levels

Padding Type	Number of retrieved images considered		
	15	30	45
Zero Padding	71.52	82.25	86.57
Periodic Padding	71.56	82.60	86.84
Symmetric Padding	71.72	82.48	86.86

TABLE 4.8: Percentage of recognition rate for different image padding

effect the retrieval accuracy of the discrete wavelet frames, since all three padding types give a very similar retrieval rate. Thus the choice of image padding is not crucial in the context of texture retrieval. However, the translation invariant property discussed in the previous section is affected by the choice of image padding. The translation invariance of discrete wavelet frames can only be achieved using periodic padding. Hence if we are to preserve the translation invariance property throughout this thesis, the periodic padding needs to be employed. The symmetric padding however might be useful in texture segmentation since periodic padding might affect the boundary coefficients if the opposite ends of an image contains different texture. Therefore the periodic padding will be used for feature extraction purpose while the symmetric padding will be used for segmentation in chapter 6.

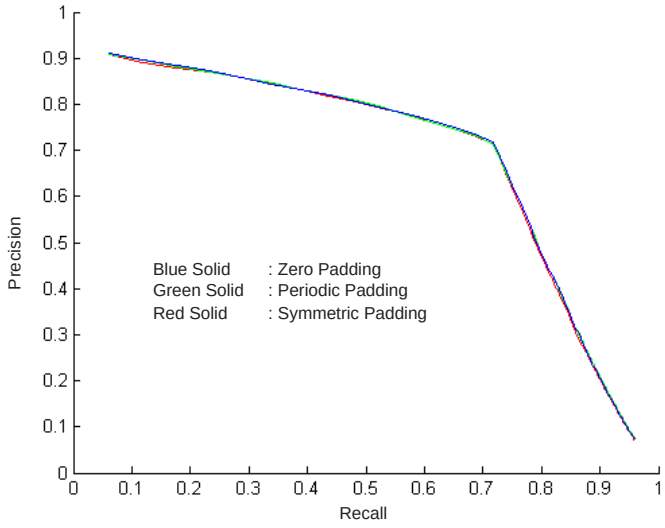


FIGURE 4.12: Precision-recall plot for different image padding

4.3.2.4 Mean Subtraction

There are several wavelet papers which suggest a mean value of the image is subtracted first before the wavelet transform or the wavelet frames are applied (78; 119). In this section, the advantage of this preprocessing stage is investigated. In theory, this preprocessing step can help in achieving invariance of the brightness value of the image. Since the LL channels of the wavelet transform and the wavelet frames is obtained by filtering with a filter which is non-zero sum between its coefficients, the resulting LL coefficients will be highly dependent on the brightness of the image. In other words, two similar texture but with a different brightness will have a different feature in the LL channels. Removing the mean from the input image is an alternative to solve this problem. Figure 4.13 shows the precision-recall plot for the retrieval process using the original image grey levels versus the retrieval process with the mean-removal pre-processing algorithm.

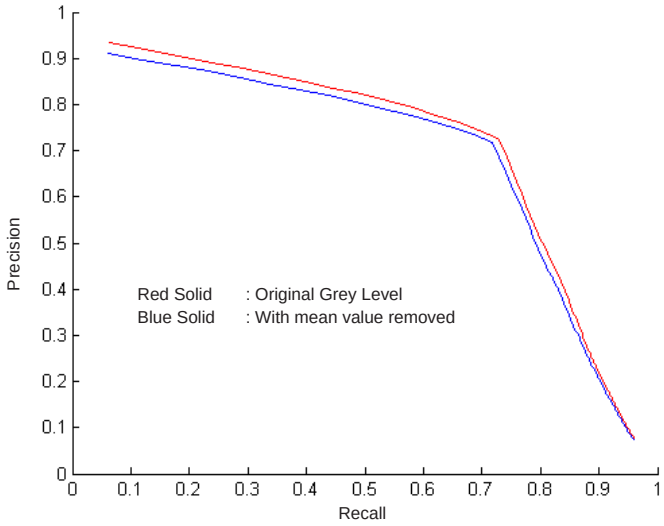


FIGURE 4.13: Precision-recall plot for feature extraction on original images vs. mean-removed images

From the figure, the mean removal does not seem to have an advantage over the ones using the original grey level, in fact it is slightly worse. However after visually inspecting the behaviour of the retrieved images, it appears that the pre-processing step helps in grouping together images with the same texture configuration, even though they are from different classes. Figure 4.14 shows the result of top 50 retrieved images of the two approaches using one of the D001 sub-images as the query. As can be seen, with mean removal, textures with vertical and horizontal stripes are grouped together, while that is not the case for the ones using the original grey levels. For this reason, and the fact that it can also solve the problem of brightness invariance, the mean removal process will be used before discrete wavelet frames is applied.

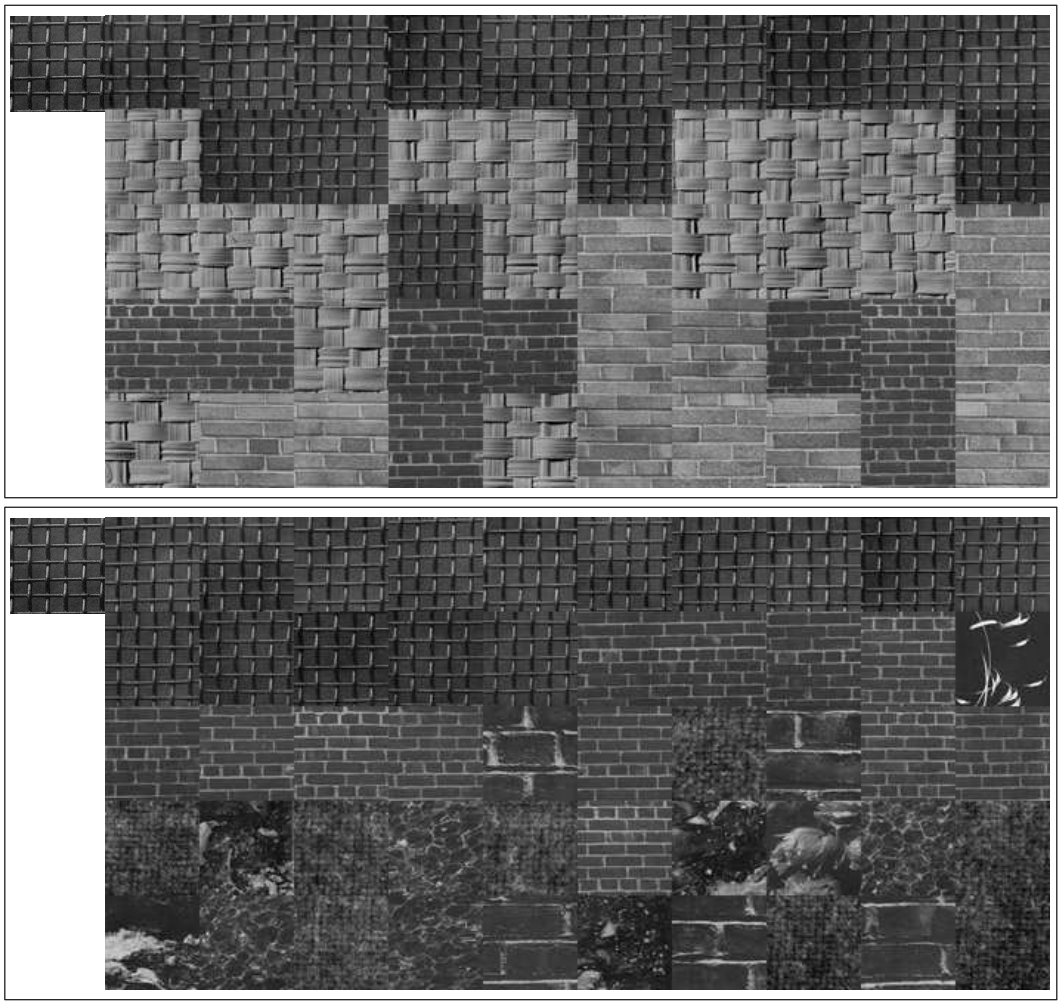


FIGURE 4.14: Top 50 retrieved images for feature extraction on (top) mean-removed images (bottom) original image. The query is located at the top left

4.3.2.5 Distance Metrics

Distance measure is an important issue in texture retrieval. The Euclidean distance is the most commonly used distance metric in texture retrieval. However since we are using wavelets, the Euclidean distance are not very suitable since the range of the individual features tend to increase dramatically with increasing resolution, i.e. for the lower frequency. This might results in the lower frequency components dominating the distance measure, hence making the higher frequency components have very little influence in texture discrimination. As an example, the mean and standard deviation of the entire Brodatz texture sets for the discrete wavelet frames features are given in table 4.9, where f_i is the individual feature component, with f_1,f_2,f_3,f_4 are the features from the 3rd level coefficients, f_5,f_6,f_7, are from 2nd level, and f_8,f_9,f_{10} are from the 1st level.

As can be seen from the table, the higher the levels, i.e. the lower the frequency channels,

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
Mean	156.58	66.45	68.31	44.73	37.62	36.44	20.93	15.81	15.18	6.42
Standard Deviation	95.01	36.33	45.97	28.29	22.21	24.72	11.73	10.01	10.11	5.600

TABLE 4.9: Mean and standard deviation of each individual feature of the DWF features, taken over the entire Brodatz texture sets

the higher and bigger the feature range. It is therefore interesting to observe whether the Euclidean distance is suitable for use with the wavelet-based features. One alternative way to reduce the domination of the lower frequency components is to normalize each individual features with the standard deviation of the corresponding features. It is also interesting to observe whether the order in the L_m distance metric contributes to the overall retrieval rate. The Euclidean and the Manhattan (L_2 and L_1 distance respectively) distance metrics will be considered. All together, four distance metrics will be evaluated, and is summarized as follows:

$$\text{Manhattan distance} : \sum_k \left| f^i(k) - f^j(k) \right| \tag{4.24}$$

$$\text{Euclidean distance} : \sum_k \left(f^i(k) - f^j(k) \right)^2 \tag{4.25}$$

$$\text{Normalized Manhattan distance} : \sum_k \left| \frac{f^i(k) - f^j(k)}{\sigma(k)} \right| \tag{4.26}$$

$$\text{Normalized Euclidean distance} : \sum_k \left(\frac{f^i(k) - f^j(k)}{\sigma(k)} \right)^2 \tag{4.27}$$

There are other distance metrics such as the Mahalanobis and the Bayesian distance metrics. However these metrics are quite complex in nature and require some statistical properties of textures. This requirement is quite unsuitable for an interactive content-based retrieval application and thus we will not be evaluating such metrics. Furthermore the normalized Euclidean distance can be classified as a simplified Mahalanobis distance, and can therefore represent the Mahalanobis distance to some extent. Figure 4.15 and table 4.10 shows the recognition rate of the four different distance metrics. From the figures, both the normalized distance give an almost similar performance and is the best among the evaluated metrics. This illustrates that the domination of the low frequency components does effect the discrimination ability of the discrete wavelet frames features. When the feature is not normalized, the Manhattan distance appeared to be better than the Euclidean distance. This is caused by the fact that the domination of the low frequency features are further amplified by the square element of the Euclidean distance metric. Without the square function, the higher frequency components can still contributes to the discrimination ability of the DWF features. Nevertheless, since the normalized Euclidean distance is slightly better than the normalized Manhattan

Distance Metric	Number of retrieved images considered		
	15	30	45
Manhattan	66.84	78.80	83.53
Euclidean	61.99	73.65	78.66
Normalized Manhattan	71.72	82.48	86.86
Normalized Euclidean	72.07	83.22	87.45

TABLE 4.10: Percentage of recognition rate for different distance metrics

distance, it will be used as the distance metric for the DWF features in this thesis.

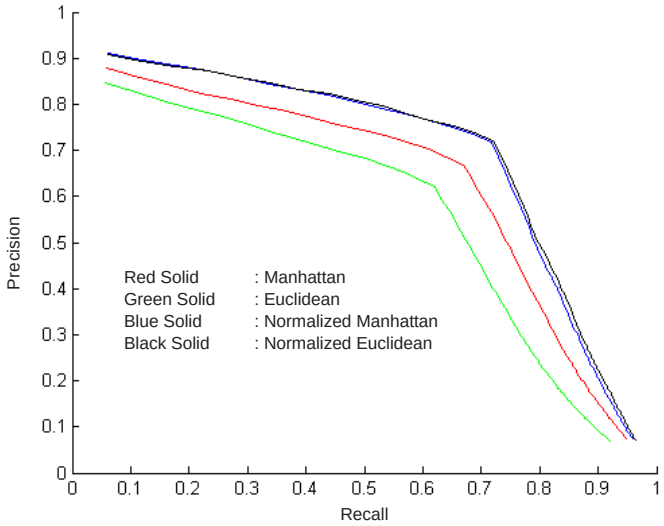


FIGURE 4.15: Precision-recall plot for different distance metrics

4.3.3 Improving the Retrieval Accuracy

Throughout the last sections, the mean energy of the coefficients in each channel is used as the feature. In this section several other statistics will be investigated for their suitability in discriminating textures. First the individual performance of the selected statistical function will be evaluated. This is followed by investigating the best combination of statistics in order to achieve the best texture retrieval performance. However in doing so, the speed constraint will always have to be observed so as not to sacrifice the low computational advantage of the discrete wavelet frames while improving the recognition rate.

4.3.3.1 Individual Functions

Ten individual features are short-listed for evaluation, which are the mean value, standard deviation value, mean energy, standard deviation energy, maximum value, mini-

mum value, maximum energy, maximum row sum energy, maximum column sum energy, and the number of zero-crossings. However, since all the channels except the *low-low* channel of the discrete wavelet frames are zero mean (the *low-low* channel will also be zero mean since we remove the mean before applying the DWF), the mean feature will provide no meaningful property. This leave us with nine functions to be considered. For an $M \times N$ image I, the nine statistical functions are defined as below:

$$\text{Standard deviation value, } \sigma = \frac{\sum_n \sum_m [I(m,n) - \mu]^2}{M \times N} = \frac{\sum_n \sum_m I(m,n)^2}{M \times N} \tag{4.28}$$

$$\text{Mean energy, } |\mu| = \frac{\sum_m \sum_n |I(m,n)|}{M \times N} \tag{4.29}$$

$$\text{Standard deviation energy, } |\sigma| = \frac{\sum_n \sum_m [I(m,n) - |\mu|]^2}{M \times N} \tag{4.30}$$

$$\text{Maximum value, } max = max\ I(m,n) \tag{4.31}$$

$$\text{Minimum value, } min = min\ I(m,n) \tag{4.32}$$

$$\text{Maximum energy, } max = max\ |I(m,n)| \tag{4.33}$$

$$\text{Maximum row sum energy, } maxrow = max \sum_n |I(m,n)| \tag{4.34}$$

$$\text{Maximum column sum energy, } maxcol = max \sum_m |I(m,n)| \tag{4.35}$$

$$\text{Average zero-crossings, ZC} = \frac{total\ number\ of\ zero\text{-}crossings}{M \times N} \tag{4.36}$$

Note that the term 'value' refers to computation on the original value of the coefficients, while the term 'energy' refers to computation on the absolute value of the coefficients. The zero crossing feature is used based on the fact that zero-crossings of a wavelet transform correspond to edges in the original image (120; 121). In some previous work (122),researchers used an edge density per unit area in order to classify texture. Therefore computing the density of zero-crossings within wavelet coefficients are similar to computing edge density in the original image. Furthermore, being a multiresolution technique, computing the zero-crossings on different wavelet scales are equivalent to computing the edge density of different edge thresholds. The zero-crossings computation is therefore believed to be a useful tool for texture discrimination.

Figure 4.16 and table 4.11 shows the performance of the nine functions. From the figure, it is quite clear that the standard deviation energy features perform best followed

closely by the standard deviation value. The mean energy and the zero-crossings feature make up the top four rank statistical features. The performance of the maximum value, minimum value, maximum energy, maximum row sum energy and maximum column sum energy do not fare very well compared to the top four functions. The pair of maximum and minimum values give a very similar performance, as do the performance of the maximum row sum energy and the maximum column sum energy. The top four functions, the standard deviation energy, the standard deviation value, the mean energy and the zero-crossings are chosen for the combination of functions experiments.

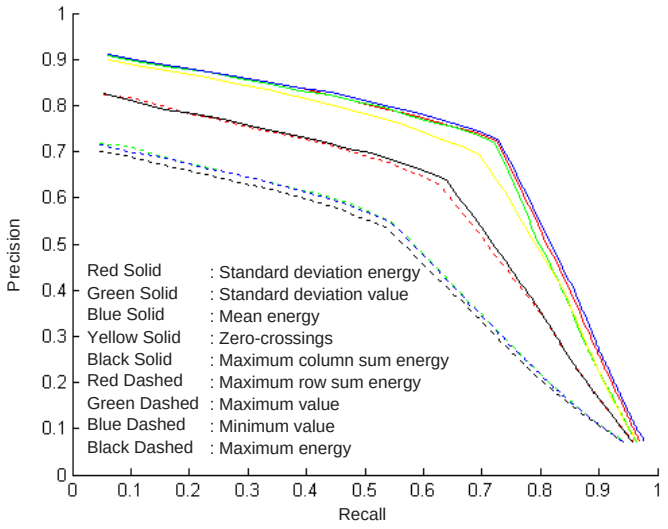


FIGURE 4.16: Precision-recall plot for nine statistical functions

Function	Number of retrieved images considered		
	15	30	45
Std deviation value	72.36	84.24	88.68
Mean energy	72.07	83.22	87.45
Std deviation energy	72.78	84.74	89.25
Max energy	53.52	68.98	76.10
Max value	54.66	69.88	76.94
Min value	54.59	69.63	76.86
Max row sum energy	62.82	77.70	83.63
Max column sum energy	63.87	78.08	83.78
Zero-crossings	69.42	83.28	87.58

TABLE 4.11: Percentage of recognition rate for nine statistical functions

4.3.3.2 Combination of Functions

From the nine listed functions in the previous section, it is logical to see the effect of combining the minimum value function with the maximum value function as well as combining the maximum row sum energy with the maximum column sum energy

function. Therefore before the top four statistical functions are examined for suitable combinations, the above two combinations will first be investigated. Figure 4.17 shows the precision-recall plot for the combined functions, along with the individual functions for comparison purpose. Both the combination functions help in improving the retrieval rate, although it is still not as high as the best individual function which is the standard deviation energy. Therefore it is impractical to used these combined functions as the texture features since the length of the features have doubled but are still inferior to the best individual function with feature length 10.

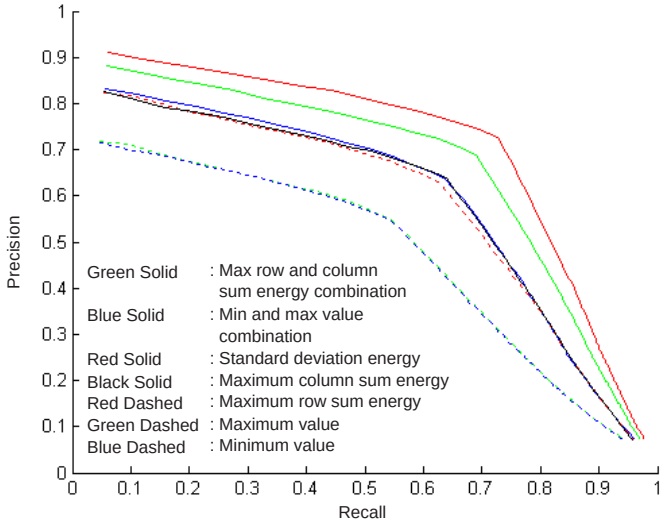


FIGURE 4.17: Precision-recall plot for two types of functions combination

Now the top four functions which are the standard deviation energy, standard deviation, mean energy and the zero-crossings will be evaluated. The following summarizes the combination of the functions:

- F_1: standard deviation energy + zero-crossings,
- F_2: standard deviation energy + mean energy,
- F_3: standard deviation energy + standard deviation value,
- F_4: zero-crossings + mean energy,
- F_5: zero-crossings + standard deviation value,
- F_6: mean energy + standard deviation value,
- F_7: standard deviation energy + zero-crossings + mean energy,
- F_8: standard deviation energy + zero-crossings + standard deviation value,

Combination Function	Number of retrieved images considered		
	15	30	45
F_1	80.67	90.84	93.62
F_2	74.59	85.73	90.02
F_3	76.29	86.93	90.46
F_4	79.68	89.97	92.99
F_5	78.68	89.30	92.25
F_6	73.20	84.52	88.65
F_7	80.80	90.58	93.56
F_8	81.18	90.83	93.70
Std dev. energy	72.78	84.74	89.25

TABLE 4.12: Percentage of recognition rate for different function combinations compared to the standard deviation energy

The first six combinations consists of 20 features while the last two combinations consists of 30 features. Figure 4.18 and table 4.12 shows the performance of the eight combined functions. The performance of the standard deviation energy alone is also plotted for comparison. From the graph, almost all combinations of function improve the retrieval accuracy significantly. Combination F_8 for example reaches 81% recognition rate for top 15 retrieved images, an increment of almost 9% over the best individual functions performance. Overall, the best combination is achieved with F_1, F_7 and F_8 combinations. It is also interesting to note that the combination of 3 functions does not have any advantage over the best combination of two functions. Therefore we can conclude that for an optimal feature extraction technique using discrete wavelet frames, 20 features are adequate to obtain a good retrieval performance. In this experiment, it is found that the combination of the standard deviation energy with the zero-crossings are the best possible DWF features.

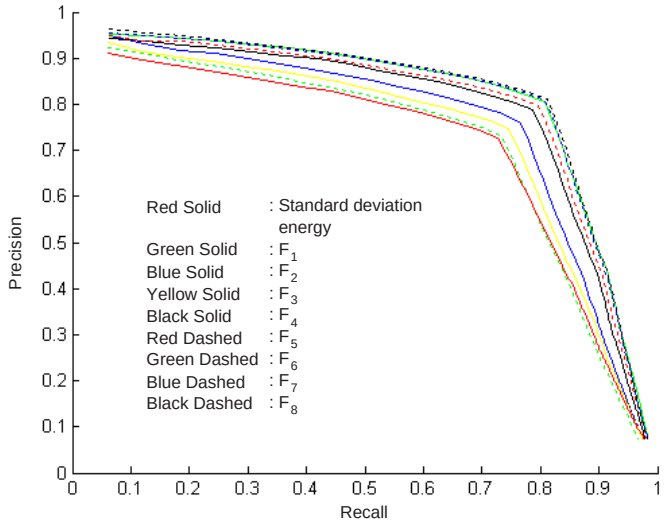


FIGURE 4.18: Precision-recall plot for eight types of functions combination

4.3.3.3 Channel Selection

The standard deviation energy and the zero-crossings feature will be used as texture features for each of the channels of the discrete wavelet frames. For a 3 level decomposition, this results in 20 features for each image. In this section the performance of individual channels will be investigated. In other words, we would like to find if using all channels are necessary to obtain the best retrieval result, or dropping some channels (hence reducing the number of features) would not cause any reduction in performance. In certain wavelet papers, the LL channels are dropped for certain reasons, while some other papers tend to drop the HH channels. In this experiment, the following channels selection will be tested:

- All channels,
- All channels except the HH channel of each level,
- All channels except the LL channel,
- Only the LH and HL channels of each level,

The two functions, standard deviation energy and the zero-crossings will first be evaluated separately, followed by a combination of the two as the following:

- S_1: Standard deviation energy of all channels,
- S_2: Standard deviation energy of only the LH and HL channels of each level,
- S_3: Standard deviation energy of all channels except the LL channel,
- S_4: Standard deviation energy of all channels except the HH channel of each level,
- Z_1: Zero-crossings of all channels,
- Z_2: Zero-crossings of only the LH and HL channels of each level,
- Z_3: Zero-crossings of all channels except the LL channel,
- Z_4: Zero-crossings of all channels except the HH channel of each level,
- C_1: $S_1 + Z_1$,
- C_2: $S_1 + Z_2$,
- C_3: $S_1 + Z_3$,
- C_4: $S_1 + Z_4$,
- C_5: $S_2 + Z_1$,

- C_6: $S_2 + Z_2$,
- C_7: $S_2 + Z_3$,
- C_8: $S_2 + Z_4$,
- C_9: $S_3 + Z_1$,
- C_{10}: $S_3 + Z_2$,
- C_{11}: $S_3 + Z_3$,
- C_{12}: $S_3 + Z_4$,
- C_{13}: $S_4 + Z_1$,
- C_{14}: $S_4 + Z_2$,
- C_{15}: $S_4 + Z_3$,
- C_{16}: $S_4 + Z_4$,

The performance of channels selection on the two functions separately is shown in Figure 4.19. On both functions, the best performance was observed when all the channels are used. The smaller the number of channels used, the poorer the retrieval performance. This is further confirmed in Figure 4.20 when among all 16 combinations of channels selection, the ones which employ all channels is found to be the best. It is then safer to include all the channels when computing both the standard deviation energy and the zero-crossings.

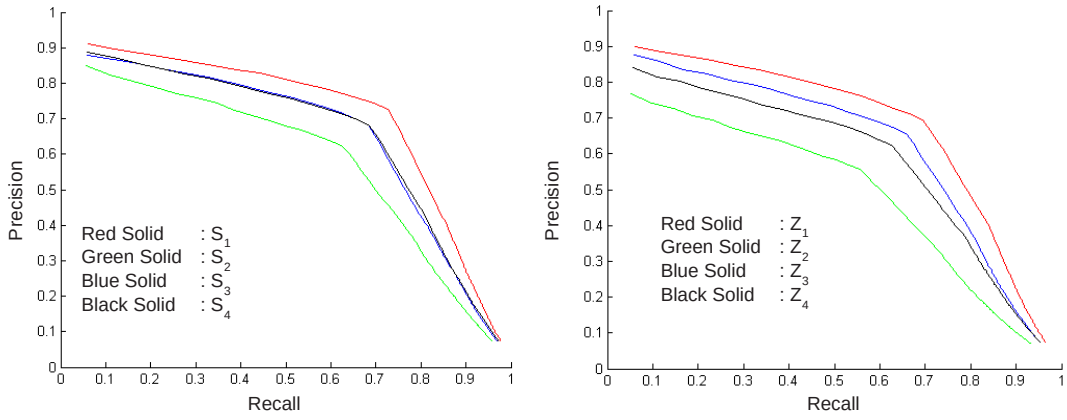


FIGURE 4.19: Precision-recall plot for different channels selection using, (left) std dev. energy, and (right) zero-crossings

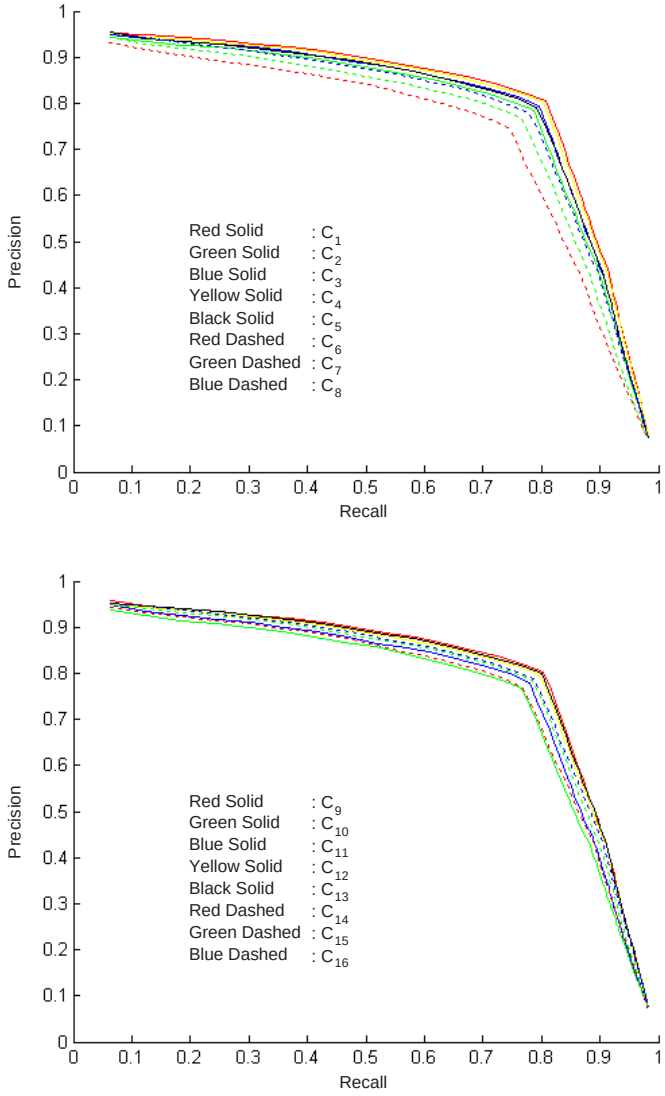


FIGURE 4.20: Precision-recall plot for different channels selection of 16 combinations

4.3.4 The Finalized DWF Texture Method

From the early experiments conducted, the discrete wavelet frames method is found to be the best texture feature method for texture retrieval in terms of accuracy and speed. The subsequent experiments investigated the best parameters and features to be associated with the discrete wavelet frames, and is summarized in table 4.13.

The above constraints resulted in a feature length of 20, and computational speed of 0.19 seconds for an image of size 128×128. The average recognition rate for the 100 homogeneous Brodatz textures on the entire database is recorded at 80.67% for $R = 15$. The recognition rate of each individual class is given in Appendix B.

Level of decomposition	3
Wavelet basis	Not crucial, but Daubechies 8-tap is chosen
Padding type	Periodic for feature extraction, symmetric for segmentation
Image pre-processing	Mean subtraction
Distance metric	Normalized Euclidean
Statistical features	Standard deviation energy, Zero-crossings
Channels selection	All channels

TABLE 4.13: Summary of the best discrete wavelet frames parameters

4.3.5 Evaluation on Colour Image Database

The discrete wavelet frames texture method described in the previous section will now be evaluated on the colour image database. To achieve this, the collection from the Vision Texture (VisTex) (123) database will be used. A total of 2672 128×128 database images are produced from the original 167 512×512 images. Each image is converted to a grey level image using the luminance function:

$$\text{Luminence, } L = 0.299R + 0.587G + 0.114B \tag{4.37}$$

where R, G and B is the red, green and blue components of the colour spaces respectively.

If the Vision Texture database is visually inspected, there are parts of the original images that are almost identical. Compared to the Brodatz texture database, it is more rigorous in distinguishing textures. With human visual discrimination, using the VisTex sub-images, it is impossible to decide which original texture to associate some of the sub-images with. This can severely distort the *true* accuracy of the texture algorithm. There are also more collections on highly inhomogeneous textures existing in the database, such as the Buildings and Paintings, compared to Brodatz textures. Figure 4.21 illustrates some of the patterns in VisTex. There is hardly any difference between Tile.0000 and Tile.0001, whereas Buildings.0005 and Painting.1.0000 are significantly uneven over the whole image. In appendix C, a list of which VisTex textures belong to the same class is presented, as well as which textures will not be included in the performance evaluation.

To obtain the true accuracy of the retrieval, the precision and recall approaches employed in the previous section need to be modified. This is because since some sub-images such as the ones from Tile.0000 and Tile.0001 are grouped together as one class, the size of each classes will no longer be the same. Some will have 16, and some will have 32. The most extreme case is within the Terrain classes where all 11 parent images are very similar, hence this class will consists of $11 \times 16 = 176$ database images. It is therefore unfair to measure the recognition rate at $R = 15$ for all classes. In order to observe the retrieval performance of the VisTex database, the following approach is employed. For a texture class of size N, we observe how many of the database images within this

FIGURE 4.21: Examples of very similar textures and highly inhomogeneous textures of the VisTex database

class is retrieved within the top $N-1$ retrieved images when one of them is used as query. For example, for a class of size 16, the recognition rate is recorded for the top 15, while for class of size 32 and 176, the recognition rate is recorded for the top 31 and 175 respectively.

The recognition rate for each VisTex class is shown in Appendix D. It was observed that the average recognition rate for the VisTex database is 68.58%. This is quite a high retrieval rate considering the level of confusion the Vistex database brought with them. The performance of standard deviation energy and the zero-crossings individually was also observed, and is recorded to be 56.55% and 53.52% respectively. This further confirms the superiority of the combined features over the individual features for VisTex database. To investigate the effectiveness of the luminence function in converting colour images into grey scale images, a selection of several colour to grey scale conversion formulae were also evaluated. The selected colour to grey scale conversions are summarized below:

- CL_1: $I = 0.333R + 0.333G + 0.333B$
- CL_2: $I = 0.405R + 0.116G + 0.133B$
- CL_3: $I = 0.145R + 0.827G + 0.627B$
- CL_4: $I = 0.596R - 0.274G - 0.322B$
- CL_5: $I = 0.211R - 0.253G + 0.312B$
- CL_6: $I = 0.500R - 0.500B$
- CL_7: $I = -0.5R + G - 0.5B$
- CL_8: $I = R$
- CL_9: $I = G$
- CL_{10}: $I = B$

Table 4.14 shows the average recognition rate for the different colour to grey scale conversion. From the table, except for a few conversion methods, all the colour to grey scale conversion approaches give quite a similar performance. We can conclude that the choice of colour to grey-scale conversion is not very critical in texture retrieval using discrete wavelet frames, and the two most commonly used approaches, the luminence and the average, CL_1 gives a comparable performance. Figure 4.22 shows some retrieval examples of the VisTex database experiments.

Conversion Type	Average Recognition Rate
Luminence, CL_0	68.58
CL_1	69.02
CL_2	68.97
CL_3	68.22
CL_4	63.77
CL_5	69.12
CL_6	64.35
CL_7	65.82
CL_8	68.18
CL_9	67.46
CL_{10}	67.00

TABLE 4.14: Average recognition rate for different colour to grey scale conversion

4.4 Chapter Summary

This chapter started with a brief description of some popular texture feature extraction methods. An experiment was then conducted to evaluate the performance of wavelet-based texture features, which are the PWT, TWT and the DWF. Several other techniques (Gabor transform, DCT, Law's texture feature, co-occurrence matrix and MRSAR) were also investigated for comparison with the wavelet-based method. From the experiment, it was found that the wavelet-based method performs quite comparably with the other method, with only the Gabor transform and the MRSAR techniques showing better retrieval performance. However the wavelet-based techniques have a very important advantage of very fast computation, hence making it the optimal choice for texture retrieval. Among the three wavelet-based techniques, the DWF was chosen for further experimentation because it is slightly better than the other two, as well as the fact that it is translationally invariant.

The next experiments were focused on the discrete wavelet frames where several important parameters were investigated for its influence in the performance of the DWF in retrieval rate. The best level of decompositions were found to be 3, as increasing the levels after this level tends to increase the computation time while not having much improvement in retrieval rate. The choice of wavelet basis for decomposition was found

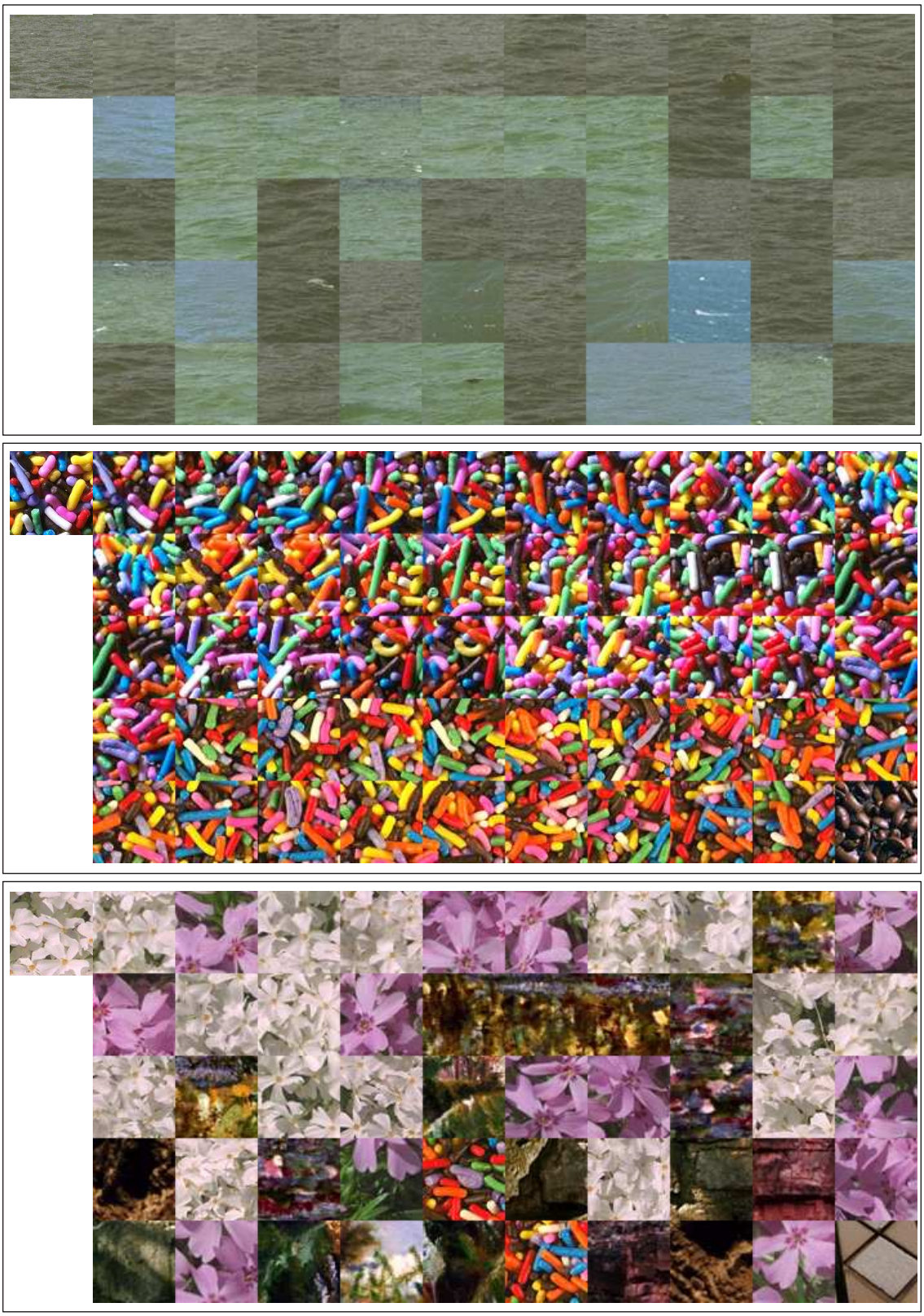


FIGURE 4.22: Examples of retrieval result for VisTex database. The query is located at the top left.

to be not critical in retrieval performance, except for the Haar wavelet which gave quite a low rate compared to the others. The Daubechies 8-tap wavelet was however chosen as the wavelet basis for later experiments as it is known to have the best localization in the spatial and frequency domain. The choice of image padding was also found to be not critical, although in order to preserve the translation invariance properties of the DWF, the periodic padding was chosen. A simple pre-processing of removing the mean of the grey level before applying the wavelet decomposition is important in texture retrieval as it helps in grouping together different textures with a rather similar configuration. Finally the best distance metric for use with the DWF features is the normalized Euclidean distance.

The next experiments focused on improving the retrieval accuracy of the DWF techniques. Several statistical functions were short-listed for evaluation and the standard deviation energy was found to have the best discrimination ability. Combining two or more statistical functions also helps in dramatically improving the retrieval accuracy, where an improvement of up to 9% was observed. However combining three or more statistical functions does not seem to have any improvement over the best combination of two functions only. Therefore the best combination of two statistical functions, which is the standard deviation energy and zero-crossings combination was chosen as the best combination. These two functions are computed from each of the DWF channels to obtain the best result. Dropping one or more channels seems to also reduce the retrieval rate.

Finally the final discrete wavelet frames features recorded a recognition rate of more than 80% for 100 Brodatz textures and almost 70% for 142 VisTex textures. In converting the colour images to grey scale, most of the conversion formulae tested give a very similar performance, indicating that the choice of conversion is not crucial. In the next chapters, the final discrete wavelet frames features presented in this chapter will be used in block -oriented decomposition as well as texture segmentation.

Chapter 5

Block Oriented Decomposition

This chapter is concerned with the effectiveness of a block-oriented decomposition in texture retrieval. A brief review of some of the popular block-oriented decomposition techniques is presented and a novel block oriented decomposition technique is proposed. Experiments to evaluate the technique are conducted on a Brodatz database as well as real museum collections.

5.1 Introduction

A common visual query to an image database system involves finding all images in the database which contain a sub-image similar to a given query. As the feature vector of a complete database image may not correctly represent its sub-images, retrieval based on comparison between the feature vectors of the query image and database images may not provide satisfactory results. Thus, image segmentation is necessary to properly implement such feature-based techniques for searching image databases (124). Effective segmentation isolates the important features of database images.

Ideally, the results generated through the process of image recognition and analysis would be used to automatically segment image content. However, as image recognition itself is still in its formative stage, investigations in this direction are still in their infancy. In terms of texture retrieval, the texture property is usually computed in local masks for localization (36). It is therefore interesting to see if texture segmentation is really needed in the *query by texture* application, or whether the local masks approach mentioned above is just as effective. As far as texture retrieval is concerned, there is no study comparing the advantages and disadvantages of the two approaches in an open image retrieval application.

This chapter will therefore focus on the block-based approach while the following chapter will focused on the texture segmentation approach. The next section briefly describes

some available methods in using block-oriented image decomposition in texture localisation within an image.

5.2 Block Oriented Decomposition Techniques

There are quite a few different approaches in using the local mask in content-based image retrieval. This includes a simple sliding mask (114; 125), the quad-, quin- and nona-tree decompositions (124; 126), and a multiscale decomposition (127), among others.

5.2.1 Sliding Windows

Sliding windows is the simplest block-oriented approach in texture localisation and is the one usually used in texture retrieval. Given an image, a collection of small to medium sized sub-images is produced by a simple image cropping procedure. The sub-images produced could be overlapping or non-overlapping, with the overlapping windows providing better localisation, but with a lot more sub-images, which could effect the speed of the feature extraction process. The size of windows usually depends on application, with large windows providing better localisation for coarse texture but risk in failing to capture small texture regions. The feature extraction process is then performed on these sub-images. During matching, the feature vector of the query image is compared with all the sub-images' feature vectors, and the sub-image corresponding to the feature vector with the least dissimilarity is taken as the region most similar to the query image. Manjunath and Ma (114) used a non-overlapping windows of size 128×128 for browsing large satellite images and air photos (about 5000×5000 pixels) with Gabor transform as the texture features. They recorded a satisfactory retrieval performance. Another example of image retrieval using sliding windows can be found in WALRUS system (125).

5.2.2 Quad-Tree Decomposition

A quad-tree is a hierarchical image decomposition structure which can provide quick access for image retrieval. A quad-tree is based on the principle of recursive decomposition of images. Each decomposition of an image segment produces four equal-sized quadrants. Figure 5.1(a) demonstrates the positions of the four quadrants, numbered 1, 2, 3 and 4, within the decomposed segment. The image decomposition process using quad-tree structure can be described recursively, with the root representing the entire image, and its children representing the decomposed segments; these, in turn, become roots for further segmental decomposition. Each internal node has exactly 4 children. The original quad-tree decomposition labels the decomposed segments white if they

(a) (b)

FIGURE 5.1: (a) Segments in quad-tree decomposition, (b) Example of quad-tree decomposition

consist of white pixels only, black if they consist of black pixels only, and grey if they consist of both black and white pixels. Further decompositions are only carried out on gray segments.

Smith and Chang (124) have presented a *query by texture* approach using quad-tree segmentation and wavelet transform. In their application of the quad-tree structure, the definition of leaf nodes is slightly changed. Before four children are generated by each parent, conditions for merging are tested. A distance threshold is computed for each child on the basis of extracted texture features. The distances in the feature space are measured from the parent node to each child. If the distance to all four children falls within the thresholds of the children, a single texture would be declared in the parent node, and no further decomposition is necessary. Otherwise, pairwise grouping of the children will then be performed. That is, if the distance between two neighbouring children falls below the thresholds of both, the children are merged as a single child. The quad-tree decomposition is then iterated on each child until the size of the smallest child reaches a certain number of pixels. Overall, the maximum number of children generated by a quad-tree decomposition is 4^i, where i is the number of the decomposition level. Figure 5.1(b) shows an example of one of the possibilities of image decomposition using the quad-tree proposed by Smith and Chang.

5.2.3 Quin-Tree Decomposition

A quin-tree is a hierarchical image decomposition structure based on a slight modification of the recursive decomposition of images used in quad-trees. Each decomposition of an image segment produces five sub-segments of equal size. In addition to the four equal-sized quadrants obtained in quad-tree decomposition, a fifth sub-segment equal in size to each quadrant is generated from the central area of the image segment. Figure 5.2 demonstrates the position of the segment, numbered 5, within the decomposed segment.

The decomposition process of a quin-tree can also be described recursively, with the root representing the entire image and its children representing the decomposed segments; these, in turn, become roots for further segmental decomposition. Each internal node has

at most five children. The strategy of quad-tree decomposition proposed by Smith and Chang (124) was used by Guo et al. (126) to guide the decomposition of sub-segments 1, 2, 3 and 4 in the quin-tree. Whether or not these sub-segments are generated determines the generation of sub-segment 5. Guo et al. listed two cases for the generation of sub-segment 5:

- *No decomposition of sub-segments 1, 2, 3 and 4:* A single texture has been declared for the segment, thus sub-segment 5 should not be generated.
- *Existence of some sub-segments among 1, 2, 3 and 4:* The segment has a heterogeneous texture, thus sub-segment 5 is generated.

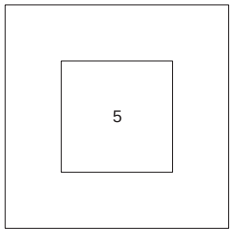


FIGURE 5.2: The fifth segment in quin-tree decomposition

Overall, the maximum number of children generated by a quin-tree decomposition is $(4^{i+1} - 1)/3$ (after eliminating redundant block segments caused by the overlapping of sub-segment 5 with sub-segments 1, 2, 3 and 4), where i is the number of decomposition level.

5.2.4 Nona-Tree Decomposition

A nona-tree is a hierarchical image decomposition structure based on a further modification of the recursive decomposition of images that is proposed in quin-trees. Each decomposition of an image segment produces nine sub-segments of equal size. In addition to the five equal segments in quin-tree, four additional segments, again of the same size, are produced from the central areas of the upper, bottom, left and right halves of the image segment. Figure 5.3 demonstrates the positions of the four segments, numbered 6, 7, 8 and 9, within the decomposed segment.

The decomposition process of a nona-tree can also be described recursively, with the root representing the entire image and its children representing the decomposed segments; these, in turn, become roots for further segmental decomposition. Each internal node has at most nine children. Similar to the definition of the leaf nodes in the quin-tree, the strategy of quad-tree decomposition proposed by Smith and Chang (124) is used by Guo et al. (126) to guide the decomposition of sub-segments 1, 2, 3 and 4 in the nona-tree. Whether or not sub-segments 1, 2, 3 and 4 are generated determines the generation of sub-segments 5, 6, 7, 8 and 9. Guo et al. listed the following cases:

- *No decomposition of sub-segments 1, 2, 3 and 4:* A single texture has been declared for the segment, thus sub-segment 5, 6, 7, 8 and 9 should not be generated.
- *No decomposition of sub-segments 1 and 2:* A single texture has been declared for the merging of sub-segments 1 and 2. Thus, sub-segment 6 should not be generated.
- *No decomposition of sub-segments 1 and 3:* A single texture has been declared for the merging of sub-segments 1 and 3. Thus, sub-segment 8 should not be generated.
- *No decomposition of sub-segments 2 and 4:* A single texture has been declared for the merging of sub-segments 2 and 4. Thus, sub-segment 9 should not be generated.
- *No decomposition of sub-segments 3 and 4:* A single texture has been declared for the merging of sub-segments 3 and 4. Thus, sub-segment 7 should not be generated.

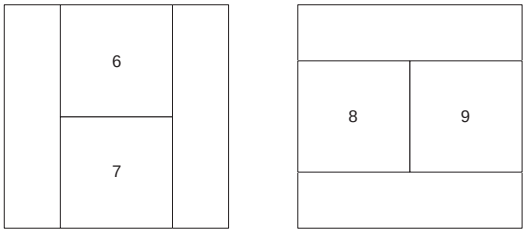


FIGURE 5.3: Additional segments in nona-tree decomposition

Those sub-segments which do not correspond to any of the above cases are generated in the tree. Overall, after eliminating redundant sub-segments caused by the overlapping of additional sub-segments, the maximum number of children generated by a nona-tree decomposition is $(2^{i+1} - 1)^2$, where i is the number of decomposition level.

5.2.5 Multiscale Image Decomposition

Multiscale image decomposition can be viewed as a multiscale version of the sliding windows described previously. In (127), Chan et al. uses a multiscale image decomposition approach in order to support colour localisation within high resolution images. Although their application is for colour features (using the colour coherence vector) rather than texture, it might also suit well for texture localisation application. The idea of the multiscale approach is to divide the database images into pyramids of patches and recording the features for each. All images are first resized to a dyadic size, and overlapping patches of size 64×64 are slid across the image. The patches are slid by an amount of half the length of the patch size. The feature vectors computed from each sub-image patch is used as feature vectors for that particular scale.

The image is then halved, resulting in images corresponding to lower resolution, and overlapping patches of the same 64×64 size are used to compute the feature vector at that scale. This process is repeated until the reduced image corresponds to a single patch, i.e. of size 64×64. Doing the decomposition this way, the patches correspond to the lowest scale representing the parent image, while patches correspond to the higher scale correspond to specific parts of the parent image. Therefore this method is a good tool in capturing both global and local features of an image. Figure 5.4 demonstrates the multiscale image decomposition process for a 256×256 image. In Chan et al.'s work, the query sub-image is also sub-divided into patches and the best match is obtained by combining the feature match scores.

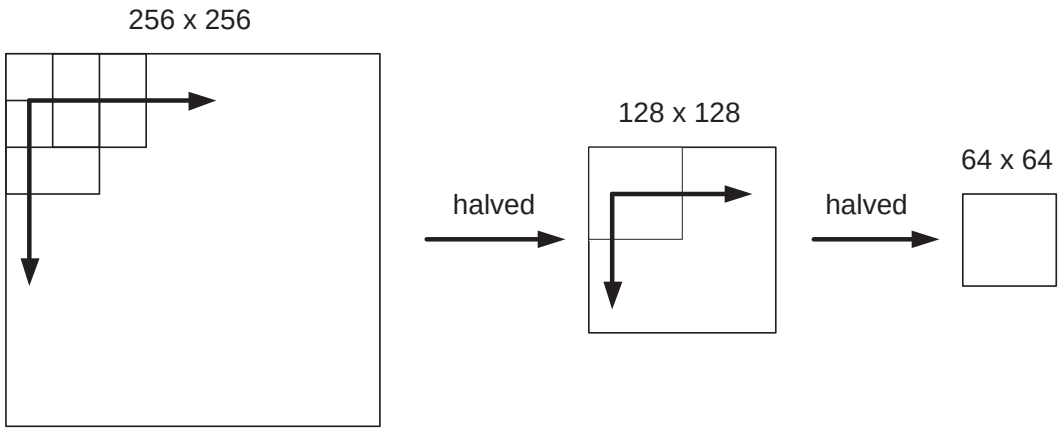


FIGURE 5.4: Multiscale image decomposition example

5.3 A Novel Block-Oriented Decomposition Approach

Based on the approaches described in the previous section, we proposed a novel block-oriented decomposition approach to be used in the content-based image retrieval of museum images (128). Our approach is based on the multiscale image decomposition method used by Chan et al. in (127). The reason for the choice is that we simply want to reduce the scale dependence of our texture feature extraction technique, the discrete wavelet frames. Moreover, while the quad-, quin-, and nona-tree approaches offer interesting methods for block-oriented decomposition, the existence of the merging threshold in the tree-based approach is a very complicated issue. A simple yet efficient variant of the multiscale approach is chosen instead. Moreover, the tree-based approaches can be viewed more or less as crude texture segmentation, therefore we think it is better to opt for a non-segmentation like approach for comparison with an actual texture segmentation approach.

Before we go to the description of our proposed algorithm, it is important to point out that the discrete wavelet frames (and transform) is not a scale invariant texture feature. Since the distribution of energy in wavelet decomposition is based on scale, texture with

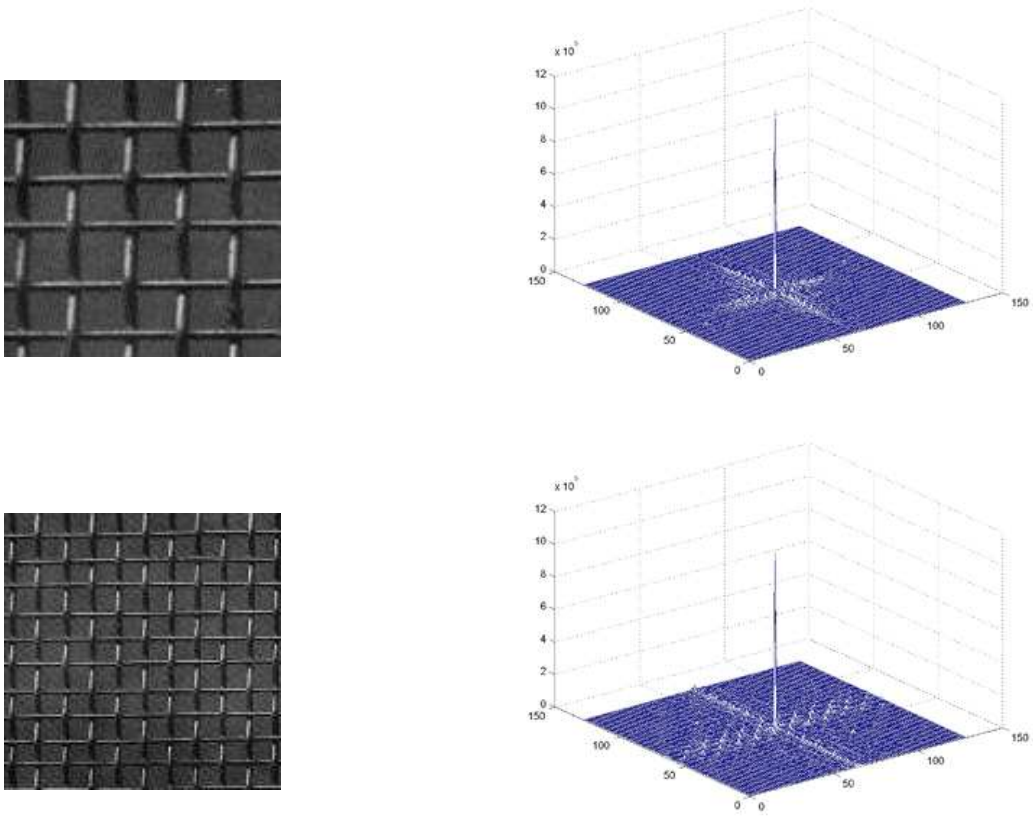


FIGURE 5.5: (top left) Coarse texture, (top right) Its frequency content, (bottom left) Fine texture, (bottom right) Its frequency content

a coarser scale will have their energy concentrated in a certain frequency range, say the LH channel of the first level decomposition. A finer scale version of the same texture will however find their energy concentrated in some other frequency range, say LH channel of the second level decomposition. Since our DWF texture features method performs a channel by channel comparison of the standard deviation of the wavelet coefficient and the number of zero-crossings within each channel, the resulting dissimilarity distance will be large due to the difference in different energy distribution. Figure 5.5 shows the distribution of frequency content for two similar textures but with different scales.

It is clear from the figure that the frequency domain of the coarse texture is more condensed in the low frequency region, while it is more spread out for the fine texture. The resulting feature vectors of the two textures will thus be very dissimilar if channel by channel comparison is performed. For this reason, we opt to use the multiscale decomposition approach in the hope that it will help in reducing the scale dependence of the texture feature. The resulting algorithm will not be totally scale invariant, but will at least reduce the scale dependence.

5.3.1 Multiscale Decomposition Algorithm

The proposed algorithm is based on the multiscale algorithm of Chan et al. where in their paper, the colour coherence vector is used to extract features from each sub-image. Because colour property does not change when re-scaling the images without maintaining the aspect ratio of the width and the height of the image, Chan et al. resize all the database images to dyadic sizes to facilitate easier image cropping and re-scaling. This means that at the lowest level, the database image will always be of size 64×64, and hence is the same size as the sub-image patch. If we are to use the multiscale approach to texture retrieval, a modification is necessary since re-scaling the image without maintaining the aspect ratio of the image's width and height tends to totally alter the properties of the underlying texture. We would like to make sure all texture properties at every level remain unaltered, except for the scales, so that the retrieved images pose a fair resemblance to the query image.

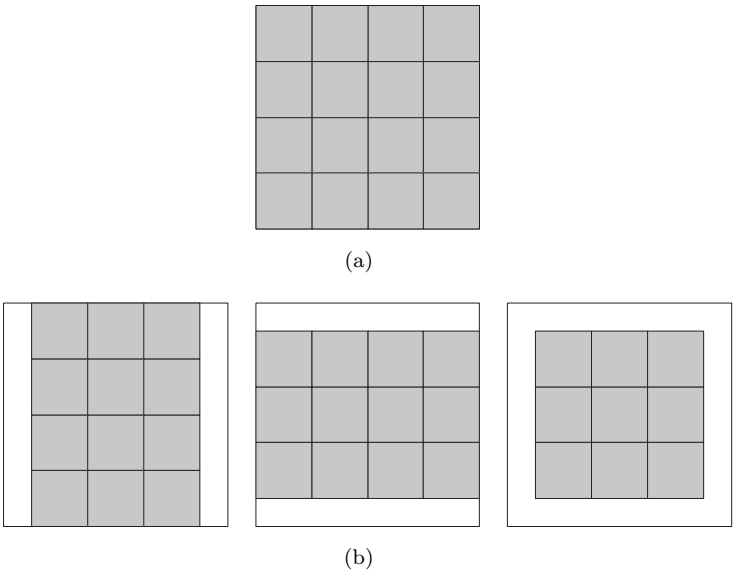
The proposed algorithm is described below. The sub-image patch used is the same as proposed by Chan et al. that is 64×64, since in real applications, we believe the query image should not be smaller than this size. Consider a texture with size 256×256. The dyadic size of the texture means that 16 sub-images will fit into the whole image at the root level as shown in Figure 5.6(a). However, there is no overlapping between sub-images, and one might argue that better localisation can be achieved by using an overlapping sub-images. Figure 5.6(b) shows that the additional 33 sub-images well placed inside the whole image, to make up a total of 49 sub-images for the overlapping case. While the overlapping approach is better localised, it requires more sub-images, which means more computation for each scale. Throughout this chapter, both overlapping and non-overlapping approaches will be investigated for their performance.

The number of sub-images, K generated for the non-overlapped and overlapped cases at any single level can be computed respectively as:

$$K = \frac{\textit{Width pixels}}{64} \times \frac{\textit{Height pixels}}{64} \tag{5.1}$$

$$K = \left(\frac{\textit{Width pixels}}{64} * 2 - 1\right) \times \left(\frac{\textit{Height pixels}}{64} * 2 - 1\right) \tag{5.2}$$

Now we will consider the case where the size of the image is not of dyadic integer, but instead any random integer value. In our multiscale image decomposition, the size of the image to be processed remains unchanged at the first level. When performing sub-image localisation, we have to allow some overlapping between sub-images even for the non-overlapped case to make sure the sub-images are evenly distributed. The number of sub-images, K for the overlapped and non-overlapped case for any single level is



(a)

(b)

FIGURE 5.6: (a) Non-overlapped sub-images, (b) Additional sub-images for the overlapped case

therefore calculated respectively as:

$$K = \left\lceil \frac{Width\ pixels}{64} \right\rceil \times \left\lceil \frac{Height\ pixels}{64} \right\rceil \tag{5.3}$$

$$K = \left(\left\lceil \frac{Width\ pixels}{64} \right\rceil * 2 - 1 \right) \times \left(\left\lceil \frac{Height\ pixels}{64} \right\rceil * 2 - 1 \right) \tag{5.4}$$

where $\lceil\rceil$ is the rounded up operator. The $\lceil\rceil$ operator ensures the sub-images are interconnected and no sections of the image will be left out. For example, consider an image with size 193×332. For the non-overlapped case, the image will contain $\lceil \frac{193}{64} \rceil = 4$ sub-images in the row direction and $\lceil \frac{332}{64} \rceil = 6$ sub-images in the column direction. For the overlapped case, the number of sub-images will be $4 \times 2 - 1 = 7$ and $6 \times 2 - 1 = 11$ in the row and column direction respectively. The amount of overlapping can be computed as below for the overlapped and non-overlapped cases respectively:

$$Amount\ of\ overlapping = 64 - \left(\frac{width\ pixels - 64}{\left\lceil \frac{width\ pixels}{64} \right\rceil - 1} \right) \tag{5.5}$$

$$Amount\ of\ overlapping = 64 - \left(\frac{width\ pixels - 64}{\left\lceil \frac{width\ pixels}{64} \right\rceil \times 2 - 2} \right) \tag{5.6}$$

Figure 5.7 shows the variation of overlapping amount with different sizes of image for the two approaches. From the figure, the minimum amount of overlapping for the non-overlapped case is 0, while the minimum amount of overlapping for the overlapped case is 32. The minimum overlapping is achieved when sizes are in multiples of 64. The

maximum overlapping is achieved when either the width or the height of the image has a size of 65 (2 sub-images, the first sub-image takes the first 64 pixels, and the second sub-image takes the last 64 pixels). Since both approaches involves overlapping of sub-images for most of the image dimension, it is necessary to rename the originally *non-overlapped* case. From this point onwards, the originally *non-overlapped* case will be referred to as *case 1 overlapping*, and the original *overlapped* case will be referred as *case 2 overlapping*.

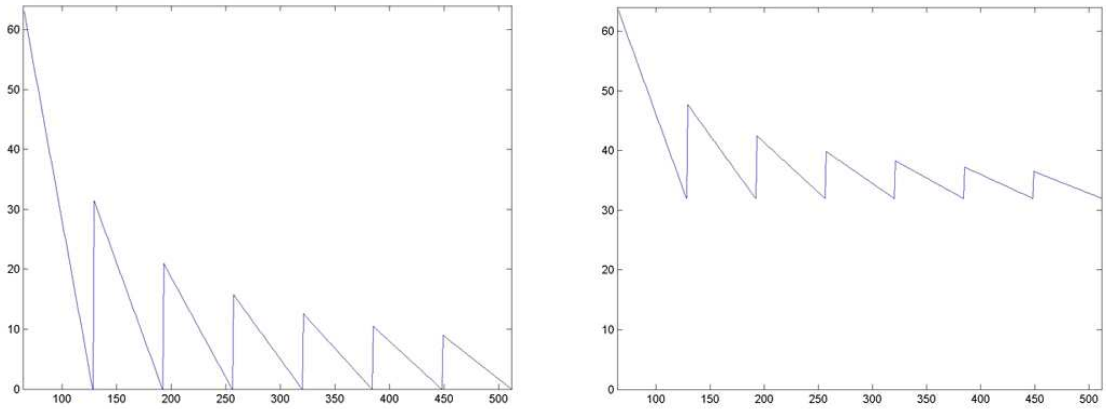


FIGURE 5.7: Amount of pixels overlapping for (left) originally non-overlapped case, and (right) overlapped case

Now that the sub-image coverage of the first scale is configured, the image re-scaling process will be now discussed. As mentioned previously, the first level of the decomposition involves the original dimension of the image to be processed. The re-scaling of the image can be described as follows. For an image with $M \times N$ dimensions, the minimum of the two dimensions, $min(M, N)$ is taken as the basis for re-scaling. Let us say the row, M is the minimum of the two dimensions. Then the image is re-scaled to the nearest dyadic integer that is smaller than M, while maintaining the aspect ratio of the width and height of the image. The sub-image decomposition described previously is then performed on the re-scaled image to get the sub-images corresponding to the second level. Starting from the second level, to obtain the parent image at the following level, the image is just re-scaled by a factor of 2. This process continues until $min(M, N)$ reached 64. For example, consider an image with size 783×556. The following illustrates the image dimensions at each level.

- First level: 783×556
- Second level: 721×512 (512 is the nearest dyadic integer smaller than $min(783, 556)$)
- Third level: 361×256
- Fourth level: 181×128
- Fifth level: 91×64

The lowest level (91×64) now consists of 2 sub-images for *case 1 overlapping* and 3 sub-images for *case 2 overlapping*. In general, for an $M \times N$ image, the number of scales can be computed as:

$$Number\ of\ scales = \left\lceil \frac{log(min(M,N))}{2log2} \right\rceil \tag{5.7}$$

Recall in chapter 4, we came across the problem of image padding in order to perform discrete wavelet frames decomposition. It was found that the periodic padding should be used if the translation invariance property is to be maintained. However, using multiscale image decomposition technique, we are dealing with image blocks and not actually separate entities. Therefore one might argue that the border information can be extracted from neighbouring image blocks by borrowing border pixels in the filtering operation. Done this way, the discrete wavelet frames decomposition and feature extraction processes are no longer independent for each spatial block, but the exchange offers an elegant solution for padding, and the order of operation can be reversed.

First, the entire image is decomposed using wavelet filtering, and then the patches can be slid across the stack of DWF coefficients in order to compute the features for each sub-image. After the image is re-scaled to appropriate size, the DWF decomposition is applied once again and so on until the lowest scale image. This means, the DWF decomposition only has to be applied k times, where k is the number of scales (once for each level), instead of applying it to each sub-image generated by the multiscale decomposition. The parent images at each scale however will need to be padded using a periodic padding. The feature extraction is also simplified using this technique, where the standard deviation and the number of zero-crossings are computed straight away within each block, like sliding a standard deviation and zero-crossings operators over a stack of images, as shown in Figure 5.8.

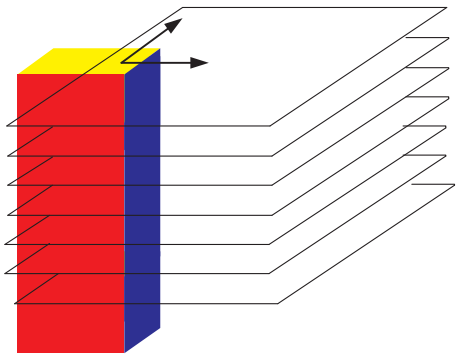


FIGURE 5.8: A cube is slid on a stack of DWF coefficient images to compute the standard deviation and zero-crossings

However, since we are subtracting the mean of the image before applying the wavelet frames decomposition (recall chapter 4), this could pose a potential problem. The purpose of subtracting the image mean is to make sure the image is zero-mean, and therefore

brightness invariance can be ensured. Applying the wavelet frames decomposition implies that the mean to be subtracted is the global mean, and not the local mean for a particular texture. As a result, not only the brightness invariance property is lost, but the retrieved images might also be inaccurate. To confirm this, we will evaluate both approaches (DWF followed by block decomposition and block decomposition followed by DWF) in the experimental section. Figures 5.9 and 5.10 show the flowchart of the two different approaches of the proposed multiscale image decomposition technique.

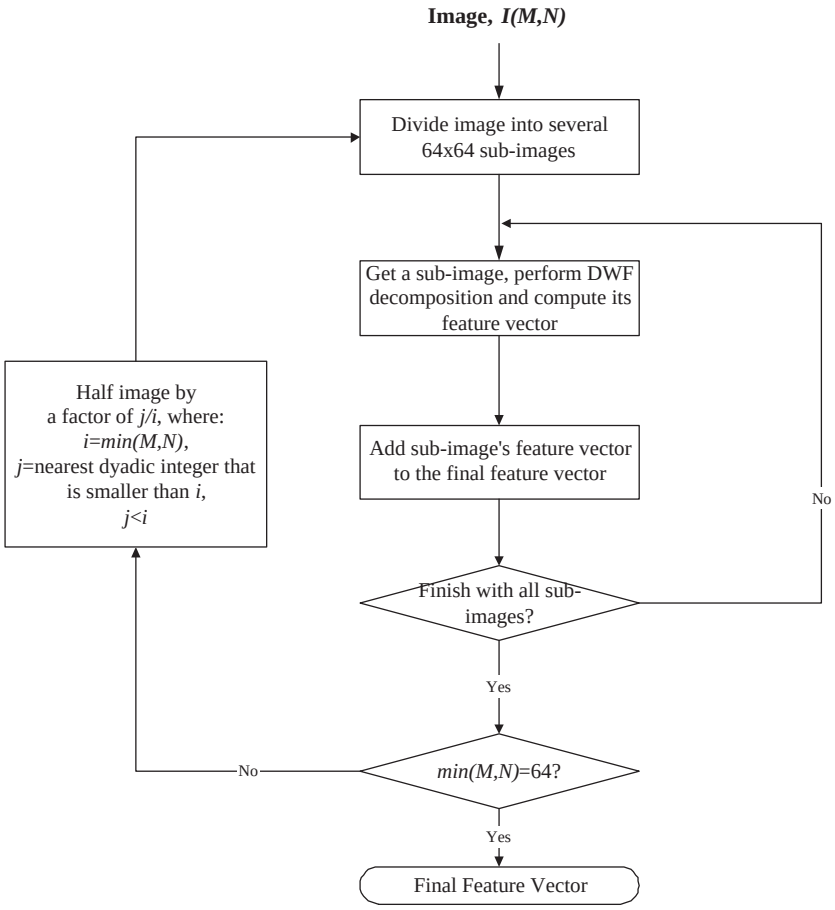


FIGURE 5.9: Flowchart of the proposed multiscale image decomposition technique (Block decomposition followed by DWF)

5.3.2 Total Number of Sub-images

Equations 5.3 and 5.4 give the number of sub-images generated at a particular scale for the *case 1* and *case 2 overlapping* respectively. The total number of sub-images generated by the multiscale algorithm for all scales can be computed by adding the number of sub-images at each scale. Figure 5.11 shows the total number of sub-images generated for *case 1* and *case 2 overlapping* for a square $M \times M$ image, with M ranging from as small as 64 up to 1024. From the figure, the number of sub-images for *case 2*

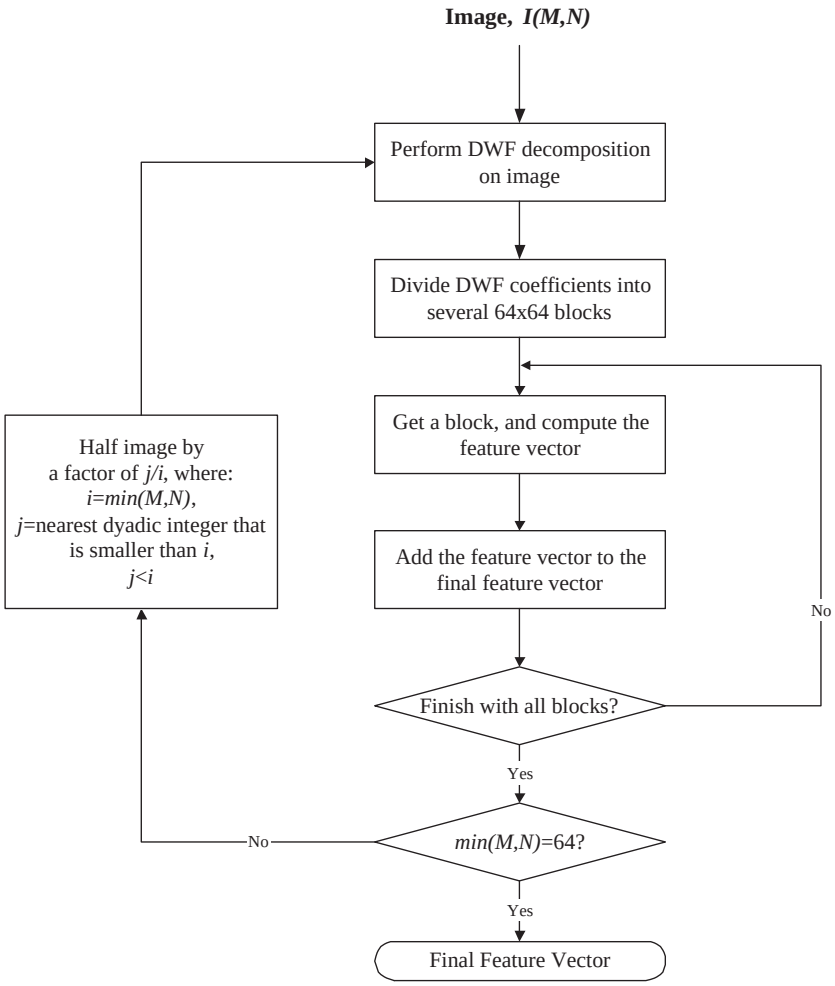


FIGURE 5.10: Flowchart of an alternative approach to multiscale decomposition (DWF followed by block decomposition)

overlapping increases almost quadratically with the increase of M over the number of sub-images of *case 1 overlapping*.

5.3.3 Sub-image Coverage

We will now discuss the overlapping coverage between a query image and the segments of a database image in instances in which the query image is similar to a sub-image of the database image. Let Q be a query image and D be a database image. We assume for simplicity that the query image is of size 64×64 and the database image is of size 128×128. Assume that D contains a sub-image d which is similar to Q and d may be anywhere in D. We now examine the degrees of coverage between the query image and the segments generated by the two approaches mentioned before, the *case 1* and *case 2 overlapping*.

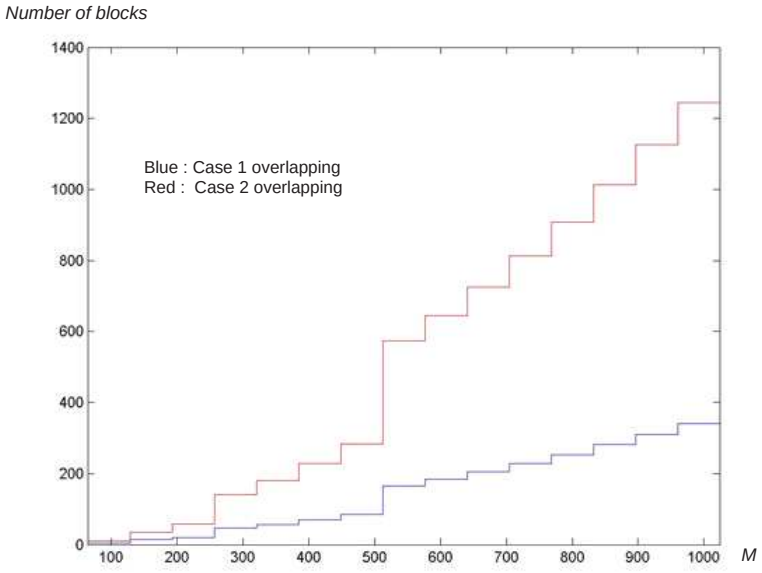


FIGURE 5.11: Number of sub-images generated

5.3.3.1 Case 1 Overlapping

For a 128×128 image, it is decomposed into 4 quadrants of size 64×64. The minimum coverage between these four quadrants of D and Q will be $\frac{1}{4}$ of Q, when d is located in the centre of D. Obviously, when d is located in other places in D, the coverage between Q and any quadrant of D will be larger than the $\frac{1}{4}$ of Q. So when the size of the database image is non-dyadic, the coverage will be larger than $\frac{1}{4}$ of Q. Figure 5.12 illustrates the situation.

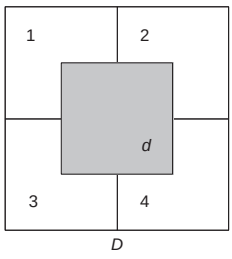


FIGURE 5.12: Minimum coverage in case 1 overlapping

5.3.3.2 Case 2 Overlapping

Case 2 overlapping approach ensures that there exists a segment in D which covers at least $\frac{9}{16}$ of Q. Figure 5.13 illustrates this situation. For a 128×128 image, it is decomposed into 9 quadrants of size 64×64. Let the query image overlap with the database image at an offset of d_1 and d_2 at each side as indicated in Figure 5.13.

In the figure, the overlap between query image Q and database image D is represented by the shaded area. Let $L = 64$ be the size of the query image, in general, as shown

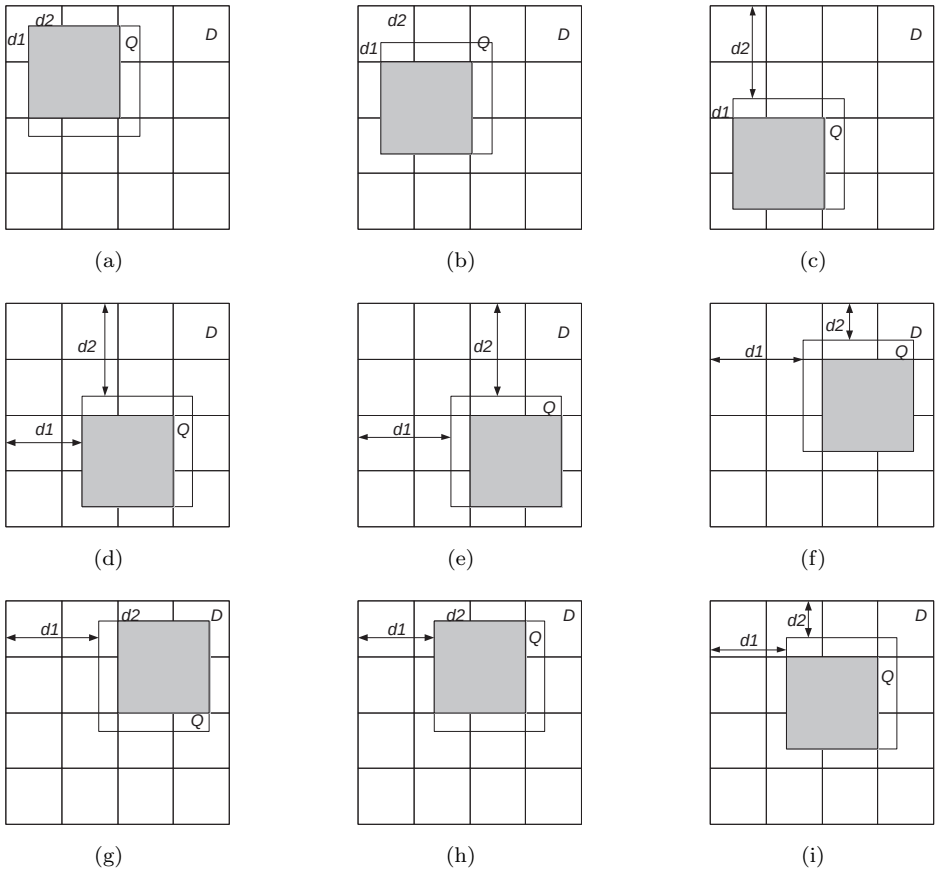


FIGURE 5.13: Covered area of query image by segments

in Figure 5.13, the shaded area A of the query image conforms to one of the following cases:

- (Figure 5.13a) $0 \leq d_1, d_2 \leq \frac{1}{4}L$: The area covered is

$$A = (L - d_1) \times (L - d_2) \geq \frac{9}{16}L^2,$$

- (Figure 5.13b) $0 \leq d_1 \leq \frac{1}{4}L$ and $\frac{1}{4}L \leq d_2 \leq \frac{3}{4}L$: The area covered is

$$\text{If } \tfrac{1}{4}L \leq d_2 \leq \tfrac{1}{2}L, \qquad A = (L - d_1) \times (\frac{1}{2}L + d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_2 \leq \tfrac{3}{4}L, \qquad A = (L - d_1) \times (\frac{3}{2}L - d_2) \geq \frac{9}{16}L^2,$$

- (Figure 5.13c) $0 \leq d_1 \leq \frac{1}{4}L$ and $\frac{3}{4}L \leq d_2 \leq L$: The area covered is

$$A = (L - d_1) \times d_2 \geq \frac{9}{16}L^2,$$

- (Figure 5.13d) $\frac{1}{4}L \leq d_1 \leq \frac{3}{4}L$ and $\frac{3}{4}L \leq d_2 \leq L$: The area covered is

$$\text{If } \tfrac{1}{4}L \leq d_1 \leq \tfrac{1}{2}L, \qquad A = (\frac{1}{2}L + d_1) \times d_2 \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_1 \leq \tfrac{3}{4}L, \qquad A = (\frac{3}{2}L - d_1) \times d_2 \geq \frac{9}{16}L^2,$$

- (Figure 5.13e) $\frac{3}{4}L \leq d_1 \leq L$ and $\frac{3}{4}L \leq d_2 \leq L$: The area covered is

$$A = d_1 \times d_2 \geq \frac{9}{16}L^2,$$

- (Figure 5.13f) $\frac{3}{4}L \leq d_1 \leq L$ and $\frac{1}{4}L \leq d_2 \leq \frac{3}{4}L$: The area covered is

$$\text{If } \tfrac{1}{4}L \leq d_2 \leq \tfrac{1}{2}L, \qquad A = (d_1) \times (\frac{1}{2}L + d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_2 \leq \tfrac{3}{4}L, \qquad A = (d_1) \times (\frac{3}{2}L - d_2) \geq \frac{9}{16}L^2,$$

- (Figure 5.13g) $\frac{3}{4}L \leq d_1 \leq L$ and $0 \leq d_2 \leq \frac{1}{4}L$: The area covered is

$$A = d_1 \times (L - d_2) \geq \frac{9}{16}L^2,$$

- (Figure 5.13h) $\frac{1}{4}L \leq d_1 \leq \frac{3}{4}L$ and $0 \leq d_2 \leq \frac{1}{4}L$: The area covered is

$$\text{If } \tfrac{1}{4}L \leq d_1 \leq \tfrac{1}{2}L, \qquad A = (\frac{1}{2}L + d_1) \times (L - d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_1 \leq \tfrac{3}{4}L, \qquad A = (\frac{3}{2}L - d_1) \times (L - d_2) \geq \frac{9}{16}L^2,$$

- (Figure 5.13i) $\frac{1}{4}L \leq d_1 \leq \frac{3}{4}L$ and $\frac{1}{4}L \leq d_2 \leq \frac{3}{4}L$: The area covered is

$$\text{If } \tfrac{1}{4}L \leq d_1 \leq \tfrac{1}{2}L \text{ and } \tfrac{1}{4}L \leq d_2 \leq \tfrac{1}{2}L, \qquad A = (\frac{1}{2}L + d_1) \times (\frac{1}{2}L + d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_1 \leq \tfrac{3}{4}L \text{ and } \tfrac{1}{4}L \leq d_2 \leq \tfrac{1}{2}L, \qquad A = (\frac{3}{2}L - d_1) \times (\frac{1}{2}L + d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{4}L \leq d_1 \leq \tfrac{1}{2}L \text{ and } \tfrac{1}{2}L \leq d_2 \leq \tfrac{3}{4}L, \qquad A = (\frac{1}{2}L + d_1) \times (\frac{3}{2}L - d_2) \geq \frac{9}{16}L^2,$$

$$\text{If } \tfrac{1}{2}L \leq d_1 \leq \tfrac{3}{4}L \text{ and } \tfrac{1}{2}L \leq d_2 \leq \tfrac{3}{4}L, \qquad A = (\frac{3}{2}L - d_1) \times (\frac{3}{2}L - d_2) \geq \frac{9}{16}L^2,$$

The above calculations are also true for different image sizes and different scales. Obviously, if the image is of non-dyadic size, the coverage will be larger than the $\frac{9}{16}$ of Q, since the overlapping between sub-images will be increased. Although *case 1 overlapping* decomposition introduces fewer segments than does a *case 2 overlapping*, the latter provides a more effective and robust platform for image retrieval. The question

of which of these approaches is better can only be answered experimentally, where an evaluation of whether it is worth generating the extra sub-images can be observed in terms of retrieval accuracy.

5.3.4 Scale Invariance

As mentioned before, the multiscale image decomposition can help in reducing the scale dependence of the discrete wavelet frames texture features. This section will look on how the multiscale decomposition approach captures different texture scales. Consider the image in Figure 5.14, which is of size 256×256. The sub-images generated by the multiscale decomposition (using *case 1 overlapping*) are also shown in the figure. There are 16 sub-images corresponding to level 1, 4 sub-images corresponding to level 2, and 1 sub-image corresponding to the lowest level. From the figure, it is clear that the texture scales are changing from coarse to fine with the increase of levels. If, let us say, a same texture but with the scale equivalent to the lowest level in Figure 5.14 is used as query, the probability that the parent image will be retrieved should be higher than the method using just only one scale, since the comparison is now performed on 3 scales, and the sub-image corresponding to the lowest scale should have the least dissimilarity compared to the sub-images corresponding to the other two scales.

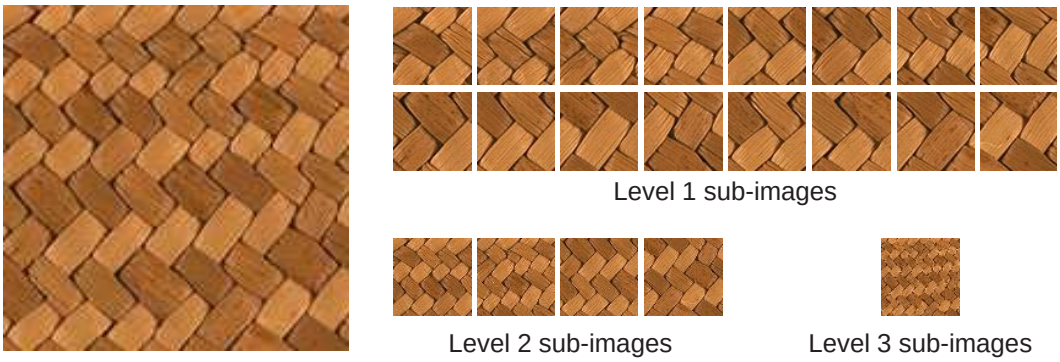


FIGURE 5.14: Example of sub-images generated for image of size 256×256

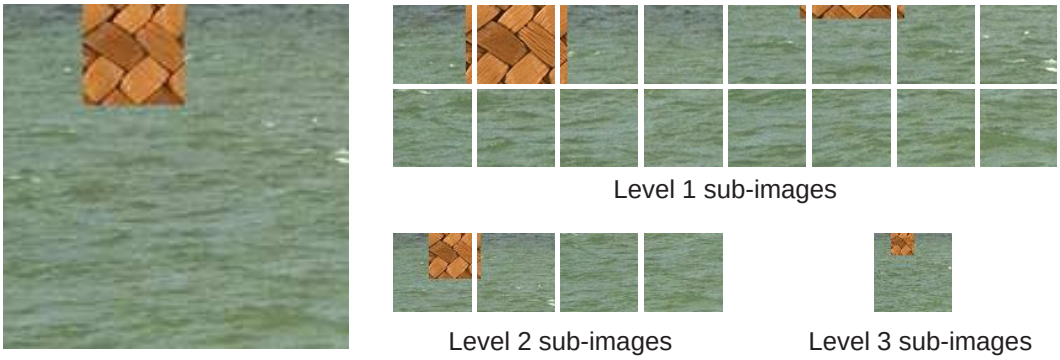


FIGURE 5.15: Another example of sub-images generated for image of size 256×256

However, the above theorem is not true for all cases. Consider the image in Figure 5.15.

The image consists of a combination of the tile texture and water, where the tile texture is only a fraction of the whole image. From the generated sub-images shown, only one of them manages to capture the whole tile region of the image, which is the second sub-image at level 1. Sub-images at level 2 and 3 fail to capture the unique tile region, thus resulting in no representative of the tile texture at that level. Hence only the original scale of the tile texture will be stored in the feature vector. The water texture however has representatives at two different scales, i.e. at level 1 and 2. We can conclude that the multiscale nature of the multiscale decomposition depends on the portion of the texture of interest within the whole image. The larger the portion, the higher the number of scales to be represented for the texture. Nevertheless, since the portion of the texture of interest in the parent image is quite small, it is unreasonable to ask for the system to process the texture at many scales, thus this small problem cannot really be considered a disadvantage of the multiscale algorithm.

5.4 Experimental Evaluation

In this section, we will discuss the effectiveness of the multiscale sub-image matching algorithm for content-based image retrieval. Experiments are conducted on three separate databases. The first database consists of dyadic size Brodatz textures. This is just to make certain evaluation easier. The second database consists of non-dyadic Brodatz textures of random sizes. Finally the third database consists of museum image collections. The evaluation takes into account several important factors, including the sensitivity of various sub-image locations within database images to which query images are compared, the size of query images and the scale of query images. The DWF technique with parameters listed in table 4.13 is used for feature extraction, and the luminence is used for colour to grey scale conversion.

5.4.1 Dyadic Size Image Database

An image data testbed was constructed from Brodatz texture images. From each 112 scanned 512×512 texture, 9 overlapping sub-images of size 256×256 are produced. Thus, there are a total of 1008 texture images in the database for the experiments. Ten Vision textures are selected and are cut-and-pasted onto a selected image from the 1008 Brodatz database to provide target textures for the multiscale algorithm. The colour Vision textures are converted to grey-scale before the cut-and-paste process. Each vision texture is pasted on 9 different database images at different locations within the images, giving a total of 90 modified database images. The Vision texture is then used as query image and the effectiveness of the multiscale algorithm is measured on the ability of the algorithm to retrieve all 9 similar textures. Figure 5.16 shows the 10 Vision textures to be used as query images.

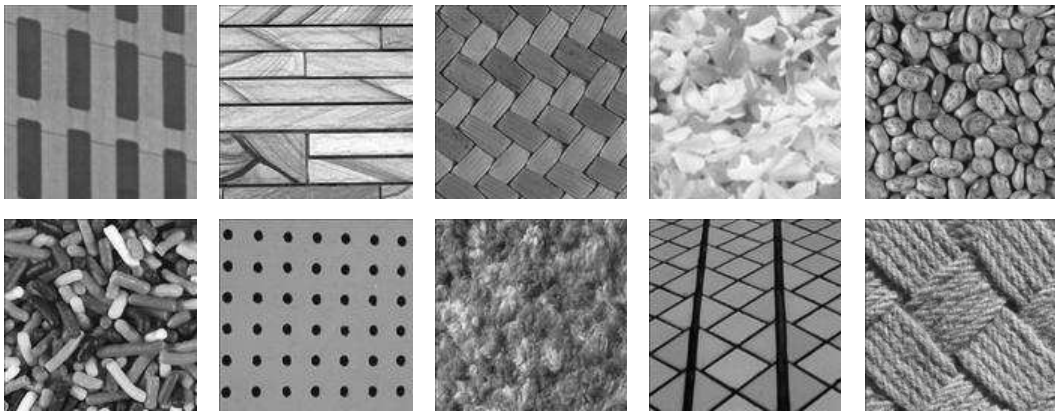


FIGURE 5.16: Vision textures used as query in the multiscale experiments

5.4.1.1 Location of the Query in Database Image

In the first experiment, the sensitivity of the sub-image locations will be tested in order to compare *case 1 overlapping* with *case 2 overlapping*. For the sake of simplicity, in this experiment, the evaluation is based on single scale only, that is the very first scale only. In other words, the scale of the query image is the same as the target sub-images, in which the algorithm should retrieve sub-images corresponding to the original scale only. The size of the query images is 64×64, while the target sub-images pasted on the database images of size 256×256 is set to 80×80 pixels. We consider two sets of experiments. The first set of experiments was performed by pasting each Vision texture onto some location in the nine overlapping regions as in Figure 5.6(b) (right most image). This experiment is to test whether *case 1 overlapping* approach is severely affected by the location of target sub-images. In the second set of experiments, each Vision texture is pasted at nine random locations within the database images.

Query Image	*Case 1 overlapping*	*Case 2 overlapping*
1	0	9
2	1	9
3	1	9
4	0	9
5	2	9
6	1	9

Query Image	*Case 1 overlapping*	*Case 2 overlapping*
7	1	9
8	0	9
9	0	9
10	2	9
Average	8.8%	100%

TABLE 5.1: Retrieval rate for fixed target location

Table 5.1 shows the result of retrieval rate for the first experiment set. The numbers given are the correctly retrieved images out of 9 possible retrievals. Clearly from the table, *case 1 overlapping* approach gave a very poor result when the coverage of sub-image is around 25% of the total query image. The retrieval rate for *case 2 overlapping* on the other hand gave a perfect 100% retrieval rate since in those specified locations, the sub-image coverage for that particular approach is 100% coverage. This experiment

shows that when the sub-image coverage of *case 1 overlapping* approach is minimum, this results in very poor retrieval rate, but the *case 2 overlapping* approach gave a very good performance. This is a huge advantage for the *case 2 overlapping* approach because when the coverage of *case 2 overlapping* is minimum ($\frac{9}{16}$ of the query image), the coverage of *case 1 overlapping* is about the same, hence in overall performance, the performance of *case 2 overlapping* should still be much better than the *case 1 overlapping* approach.

Query Image	*Case 1 overlapping*	*Case 2 overlapping*
1	5	7
2	2	6
3	4	6
4	2	4
5	6	8
6	2	6

Query Image	*Case 1 overlapping*	*Case 2 overlapping*
7	2	9
8	5	8
9	5	9
10	3	8
Average	40 %	78.88%

TABLE 5.2: Retrieval rate for random target location

This is further confirmed in the second set of experiments. Using a randomly pasted target sub-image, the retrieval results are shown in table 5.2. From the table, the performance of *case 2 overlapping* is almost double the retrieval rate for *case 1 overlapping* approach. We can conclude that *case 1 overlapping*, although it has a much lower computational intensity, is very far behind *case 2 overlapping* in terms of retrieval accuracy. Figure 5.17 shows three examples of retrieval result on randomly pasted target using the *case 2 overlapping* approach. The first two examples are the best recorded result (query image 7 and 9) while the last example shows the worst recorded result (query image 4). The box within the image shows part of the image found to be the most similar to the query.

5.4.1.2 Scale of the Query

This experiment is intended to test the multiscale nature of the proposed algorithm. The exact same set of database images as in the very first experiment (target is located at nine overlapping region) is used in this experiment, but with different scales of query images. Five different scales are tested for the query images. The size of the query image however remains the same at 64×64 pixels. The five different scales query images are produced from the original Vision texture image by appropriate resizing of the original images. Figure 5.18 shows the 5 different image scales of each query image. The multiscale algorithm used is *case 2 overlapping*, as it is the much better approach, as suggested in the last experiment.

Table 5.3 shows the retrieval result of the experiment. From the table, it is clear that as the scale of the query images changes further from the original scale, the retrieval rate drops dramatically. Using query images with the same scale as the target sub-images,

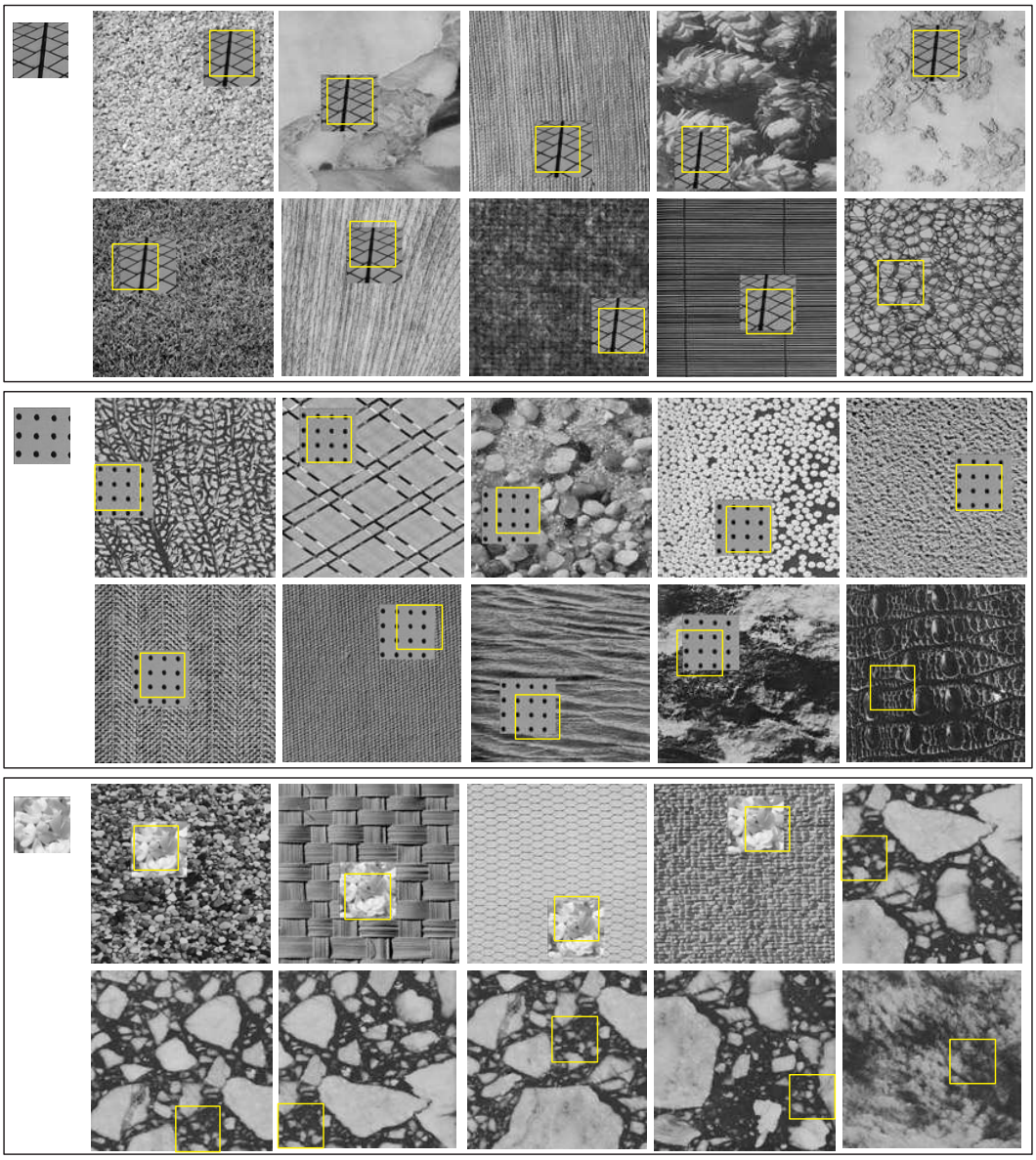


FIGURE 5.17: Example of retrieval result of multiscale matching algorithm for dyadic database

100% retrieval rate is recorded, while using query images of around twice the scale of the target sub-images (scale 5), a very poor 5.5% retrieval rate is recorded. However, note that in the database, the target sub-images is only of size 80×80 pixels. Therefore, when the image is re-scaled to the next resolution, there are no blocks that manage to capture the homogeneous target sub-images. The 80×80 target sub-images will be re-scaled to 40×40 pixels, hence the 64×64 block capturing the target region will also consists of another texture. If the size of the target images is increased, the likelihood that the 64×64 block will capture the homogeneous target region will be much better. To confirm this, another two sets of experiments are conducted. In these experiments, the 80×80 pixels target region is replaced by a much bigger target. We experimented with 140×140 and 200×200 pasted sub-images. Note that only the size of the sub-image

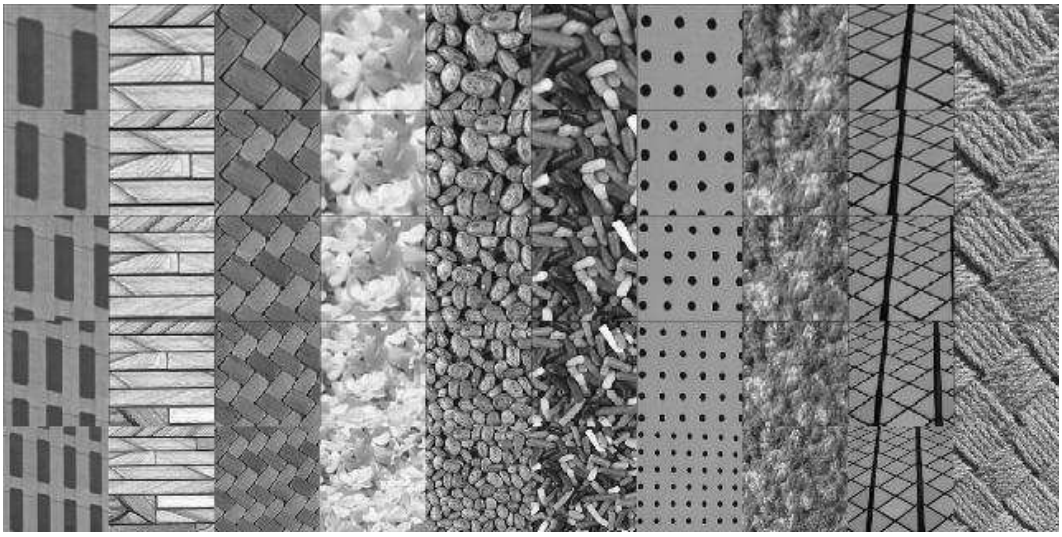


FIGURE 5.18: 5 different scales of query images

Query Image	Scale 1 (Original)	Scale 2	Scale 3	Scale 4	Scale 5
1	9	9	9	0	1
2	9	9	3	3	3
3	9	9	0	0	0
4	9	0	0	0	0
5	9	9	0	0	0
6	9	5	1	0	0
7	9	5	1	0	0
8	9	0	0	0	0
9	9	9	3	0	0
10	9	9	0	1	1
Average	100%	71.1%	18.9%	4.4%	5.5%

TABLE 5.3: Retrieval rate for different scales of query images

is different, the scale of the target remains the same.

Tables 5.4 and 5.5 shows the retrieval result of the respective experiments. From the tables, the retrieval results using different scales are very much improved, especially using scale 3, 4 and 5. This confirms our previous assumption that the multiscale algorithm works better if the proportion of the target texture is of appropriate size. The bigger the proportion, the better the multiscale algorithm works in reducing the scale dependence of the texture feature. The difference will be more obvious if the database images used are bigger than 256×256 since more scales will be involved. From tables 5.4 and 5.5, scale 5 especially gives a very good retrieval result. This is probably because scale 5 is very close to half of the original scale of the query image, which makes the feature vector of the query image of scale 5 very close to the feature vector of the second level of the multiscale. A re-scaled texture of factor 1.4, 2.4, 3.4 and so on is closer to the multiscale features at level 1, 2, 3 and so on respectively. Without the multiscale feature, only

Query Image	Scale 1 (Original)	Scale 2	Scale 3	Scale 4	Scale 5
1	9	9	9	1	9
2	7	9	6	3	9
3	9	9	5	6	9
4	9	4	0	0	9
5	9	9	9	0	9
6	8	7	2	0	9
7	9	5	0	0	9
8	9	2	0	8	9
9	9	8	3	8	9
10	9	9	0	9	9
Average	96.7%	78.8%	37.7%	38.9%	100%

TABLE 5.4: Retrieval rate for different scales of query images, with target region increased to 140×140

Query Image	Scale 1 (Original)	Scale 2	Scale 3	Scale 4	Scale 5
1	9	9	9	1	9
2	9	9	6	4	9
3	9	9	7	5	9
4	9	3	0	0	6
5	9	9	9	2	9
6	9	9	2	1	9
7	9	9	1	0	9
8	9	3	0	8	9
9	9	9	8	7	7
10	9	9	0	9	9
Average	100%	86.7%	46.7%	41.1%	94.4%

TABLE 5.5: Retrieval rate for different scales of query images, with target region increased to 200×200

the original scale is used for comparison, hence the bigger the re-scale factor, the higher the dissimilarity to the original texture. Therefore, assuming the target texture is of appropriate size, the multiscale algorithm helps in reducing the scale dependence of the discrete wavelet frames texture feature. Without the multiscale feature, textures with different scales will be much harder to match and retrieve.

Figure 5.19 shows an example of retrieval result of different scales using query image 9 (best result). The target region size is 140×140. We can see that as the scale of the query changed, the sub-images corresponding to different scales are retrieved.

5.4.1.3 Size of the Query Images

In the previous experiments, all the query images used were of size 64×64. This section will examine whether the size of the query images is crucial to the retrieval result.

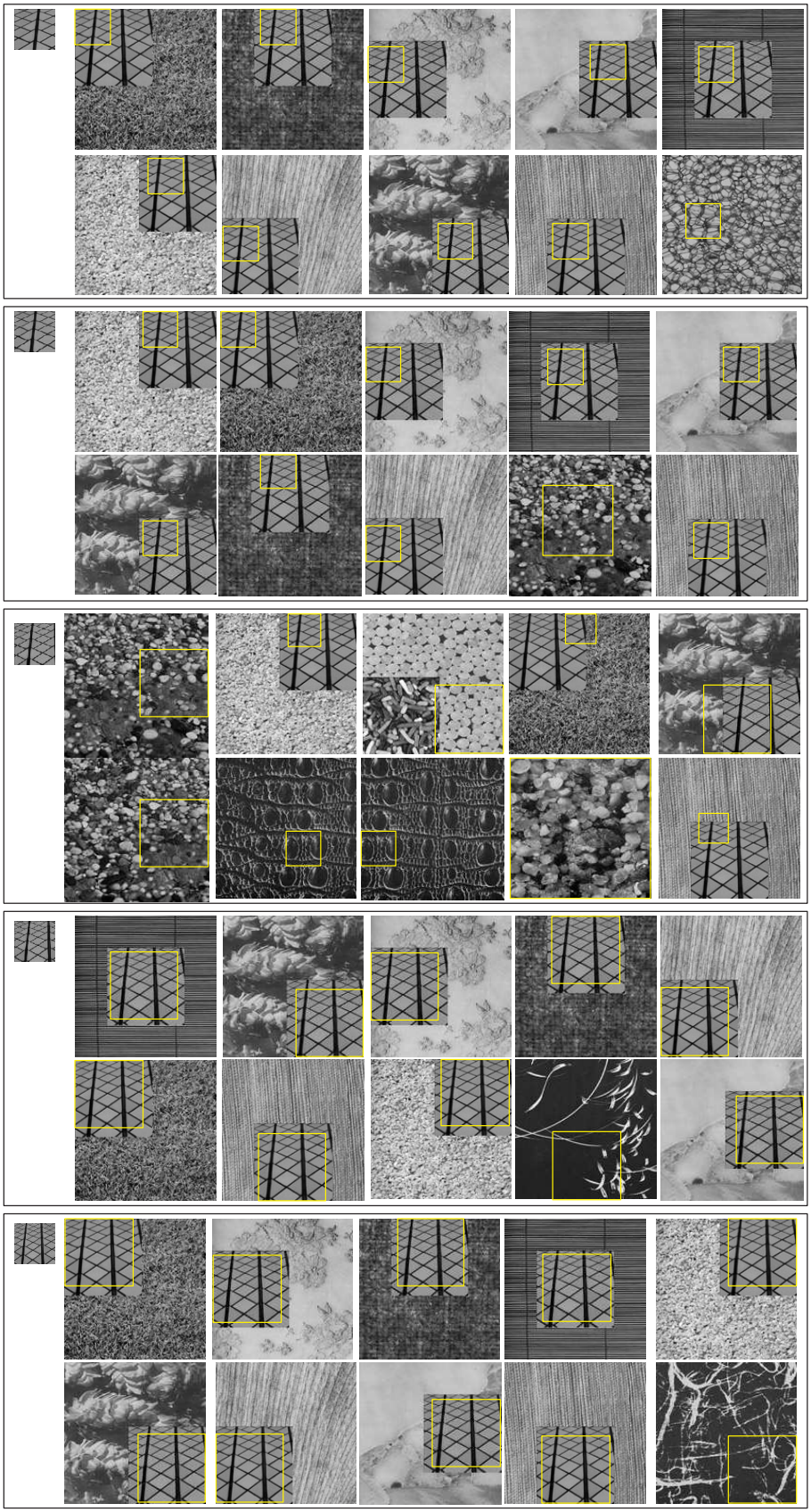


FIGURE 5.19: Example of retrieval result of multiscale matching algorithm for different scales of query images

However, it is important not to confuse this with the scale of the query images which has already been investigated in the last section. What we mean by size is the proportion of the query image itself. When we change the size of the query image from 64×64 to say, 128×128, the scale of the query image remains the same. As an example, consider an image which has a large homogeneous texture region within it. If we want to use the textured region in the image as query, we can simply crop part of the texture region and compute its feature vector. We can make a large rectangle crop or just a small rectangle crop, as long as the texture is well represented. This is what we mean by the size of query image. We would like to know if there are any differences if only a small rectangular texture is used instead of a large rectangular texture block. Theoretically, the feature vector of the two textures should not be significantly different since the standard deviation and zero-crossings computation is averaged over the total number of pixels, hence the retrieval result should not be affected much. However to confirm this, a new set of experiment is conducted. The database used is the same as the one used in the location of target experiment, where the target images are pasted on nine different overlapping regions. The query images used are of sizes 48×48, 64×64, 96×96, 128×128 and 150×150. Figure 5.20 shows the query images for the experiment.

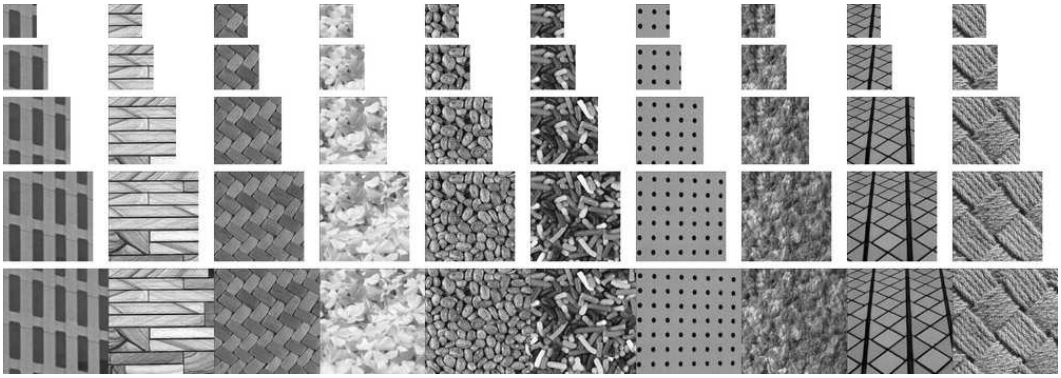


FIGURE 5.20: Query images used for experiment on the size of query images

Query Image	48×48	64×64	96×96	128×128	150×150
1	9	9	9	9	9
2	9	9	9	9	9
3	9	9	9	9	9
4	9	9	9	9	9
5	9	9	9	9	9
6	1	9	9	9	9
7	9	9	9	9	9
8	1	9	9	0	0
9	9	9	9	2	2
10	9	9	9	9	9
Average	82.2%	100%	100%	82.2%	82.2%

TABLE 5.6: Retrieval rate for different sizes of query images

Table 5.6 shows the retrieval result for different sizes of query images. The performance of different query image sizes does not seem to be very different, except for some query images (query 6, 8 and 9). Although the retrieval accuracy for query 6 using query size 48×48 is poor, it was observed that most of the top ten retrieved images consist of visually similar textures from the Brodatz collection. The ten target images are not far down the ranking, hence the retrieval result in general is still good. As for queries 8 and 9, the reason for poor results using sizes 128×128 and 150×150 is probably because as the size increases, the textures tend to vary a little bit. For example in query 8, we can see for query size 150×150 that the right hand side of the texture is darker than the left hand side, thus the feature vector produced might be a bit different from the ones produced from a 64×64 query image (which is more homogeneous). The same can be said for query 9. Hence in general we can conclude that given the properties of the texture does not change very much, the query image can take any size, which is an advantage if the CBIR system supports image cropping to provide query textures.

5.4.1.4 Decomposition Followed by DWF vs. DWF Followed by Decomposition

As mentioned previously, there are two approaches in utilizing the multiscale approach. The first approach performs the image decomposition first followed by DWF and is summarized in Figure 5.9. The second approach performs the DWF on the whole image first, followed by decomposition and is summarized in Figure 5.10. In this section an experiment is conducted to test which of these approaches is better. The database used is the one used in the experiment on location of target images, where each target image is pasted randomly on 9 selected database images. The query images are of the same scale as the target images and of size 64×64. Table 5.7 shows the retrieval result of the two approaches.

Query Image	Decomp. then DWF	DWF then decomp.	Query Image	Decomp. then DWF	DWF then decomp.
1	7	6	7	9	8
2	6	3	8	8	4
3	6	3	9	9	3
4	4	1	10	8	6
5	8	6	Average	78.9%	48.9%
6	6	4			

TABLE 5.7: Retrieval rate for different multiscale approaches

It is obvious from the table that the DWF followed by the decomposition approach does not give a good retrieval rate. As discussed previously, this is most probably because of the brightness invariance properties of the proposed multiscale algorithm, where the mean of the image was first subtracted from the image before the DWF is applied. By subtracting the mean of the whole image, only the parent image will have zero-mean,

while the texture regions of interest will not be zero-mean. When the feature vectors of these non-zero-mean texture regions are compared to the feature vector of the query texture which is zero-mean, the dissimilarity will be larger due to the difference in pixel histogram. In this thesis we will therefore stick to the original approach which is to perform image decomposition first, and then followed by DWF.

5.4.2 Arbitrary Size Images Database

Now that important measurements of the multiscale algorithm have been evaluated using a dyadic size image database, the multiscale algorithm will be tested on an arbitrary size image database. The procedure is not much different from previous experiments. From each 512×512 scanned Brodatz texture image, 9 randomly sized images are produced, which can be of any height, width and location within the parent images. This results in 1008 random size images in the database. Then for each 10 Vision textures, target images of size 80×80 are randomly pasted onto 9 different database images, resulting in 90 modified database images consisting of target textures. Each of the ten Vision textures (of size 64×64) is used as query for the retrieval experiment. Table 5.8 shows the retrieval rate for each of the ten Vision textures.

Query Image	Retrieved Target Images
1	6
2	9
3	7
4	6
5	9
6	6

Query Image	Retrieved Target Images
7	9
8	9
9	9
10	9
Average	87.8%

TABLE 5.8: Retrieval rate for random size image database

An average of 87.8% accuracy is reported from the experiment. If we compare this rate with the rate using dyadic image sizes (table 5.7), where the retrieval rate is 78.9%, the difference is quite significant. This is mainly due to the fact that, by using randomly sized images, we increase the coverage of sub-images in the multiscale algorithm, hence resulting in much better localisation. Recall that the minimum coverage of the multiscale method is $\frac{9}{16}$. This minimum coverage is achieved only when the image is of dyadic size. For images that are not of dyadic size, the sub-image coverage will be bigger than $\frac{9}{16}$. All the results obtained from previous experiments using dyadic image database actually gives the worst case scenario for each experiment. If the database is not restricted to dyadic image size, the retrieval rate will be better. Figure 5.21 shows some retrieval results using a non-dyadic image database, ranked from left to right, top to bottom. In order to accommodate the figure, images shown are resized not by the same factor.

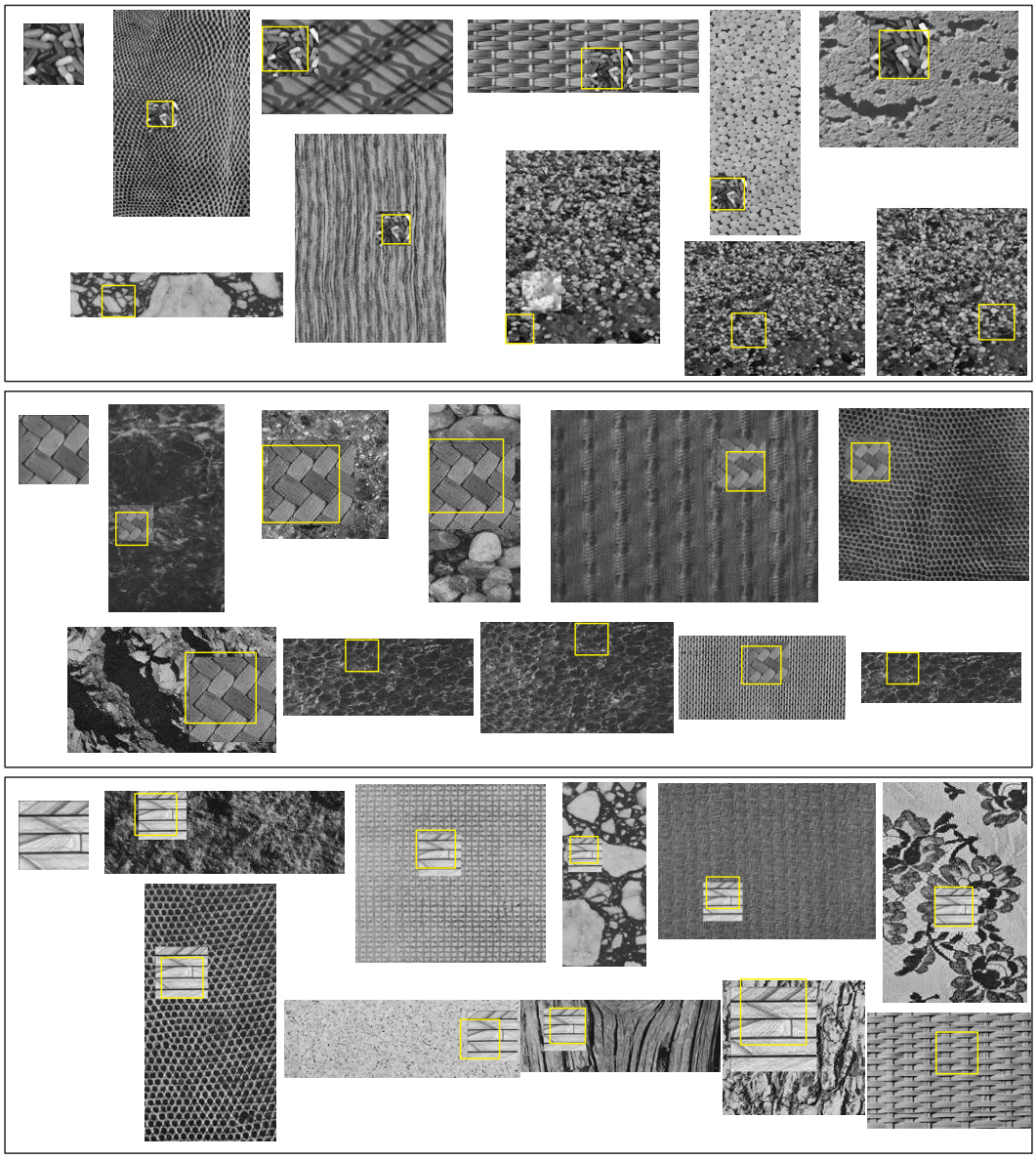


FIGURE 5.21: Example of retrieval results using non-dyadic image database

5.4.3 Museum Image Collection

In this section, the multiscale algorithm will be tested on a real database of museum images. A total of 1106 museum images of size up to more than 1000×1000 are selected from the whole museum collection available from the research group. The reason we experimented on such a small database is because at this point, we only want to investigate the suitability of the proposed method on museum image collections. Once it is found that the method is indeed suitable, then the technique will be evaluated on much bigger collection in chapter 7, together with the segmentation-based approach as well as the QBLI algorithm.

The museum collections consist of a range of museum images, from paintings to images

of interesting objects. Since we do not know how many similar textures there might be in the museum database, it is impossible to provide the retrieval rate for each query image. Therefore in this experiment, the evaluation will be based on visual inspection of the retrieval results provided. Figure 5.22 shows some retrieval results from the experiment. From the results, the multiscale algorithm does manage to retrieve visually similar texture of real museum collections. In the first example, the query is a flowery texture, and the retrieved results are all visually similar to the query images. In the second example, the query is a dotted pattern of a fabric, and the retrieved images also consists of visually similar texture, although one or two retrieved textures are not very similar. But still the top ten matches consist of mostly similar textures. In the third example, the query is a finer scale version of the second example query image, which consists of a stripe pattern. The first five retrieved images are all stripes texture with different colours. Since there are no more similar textures to the query image in the database, the next five textures results in not so similar textures. Because of a limitation in space, it is impossible to show all the results from different query images, but from the 3 examples shown, the multiscale algorithm is seen to be very useful in texture retrieval of museum collections.

5.5 Chapter Summary

The chapter starts by describing some popular methods for texture retrieval using block-based techniques. These include the sliding windows, the tree-based method and the multiscale sub-image matching. After careful study, the multiscale sub-image matching was found to be the most suitable for our application, partly because of the multiscale property of the method, which can be use to reduce scale dependence of the discrete wavelet frames texture features. Some modifications are proposed for the multiscale algorithm in order to make it suitable for texture retrieval. The modifications include the scaling factor for different scales, as well as the positioning of the blocks within the entire image of interest.

Several experiments were carried in order to evaluate the performance of the multiscale algorithm. In the first experiment, it was found that *case 2 overlapping* is far better than *case 1 overlapping*, hence making the extra sub-images of the former very worthwhile. The second experiment suggests that the multiscale algorithm was in fact a very good method in achieving scale invariant texture retrieval, although it is not perfectly scale invariant. Nonetheless, the algorithm at least helps in reducing the scale dependence of the texture features. It was also found that the size of the query images has little effect on retrieval result, making it a robust technique. The subsequent experiment suggests that the idea of performing the discrete wavelet frames decomposition first before image decomposition is quite poor, where the retrieval accuracy is very poor compared to the original approach of performing the image decomposition first before applying discrete

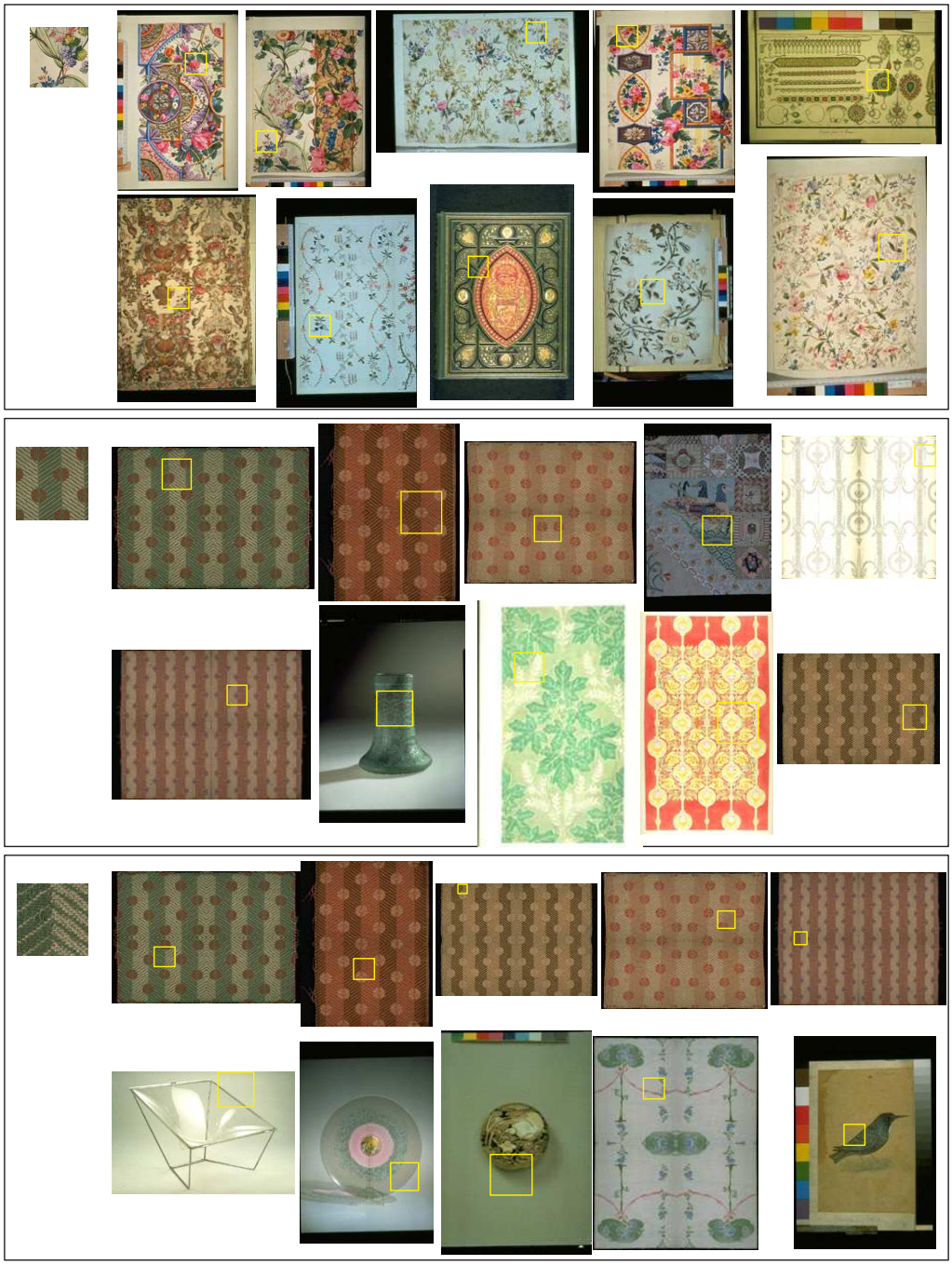


FIGURE 5.22: Example of retrieval results of real museum collections

wavelet frames. Finally the multiscale algorithm can also be used for non-dyadic image database, where good retrieval results were observed in Brodatz database as well as real database of museum image collections.

Chapter 6

Automatic Texture Segmentation

Having evaluated the block-based texture retrieval approach, the attention is now switched to segmentation-based texture retrieval. In this chapter, a brief review on texture segmentation is presented, before a novel automatic texture segmentation algorithm is developed. The algorithm is tested on several textured images including composite texture images, synthetic texture images, real scene images as well as our main source of images, the museum images of various kinds. An extension to the automatic texture segmentation, a texture identifier is also introduced in order to be integrated into a retrieval system, providing another approach to texture retrieval.

6.1 Introduction

Texture segmentation deals with identification of regions where distinct textures exist, so that further analysis can be done on the respective texture regions alone. As far as this thesis is concerned, there are three types of texture segmentation, which are:

- Supervised segmentation: This type of segmentation assumes prior knowledge of the types of the textures which exist within the image
- Unsupervised segmentation: This type of segmentation does not assumes any prior knowledge of the types of textures, but it still needs to know how many textures there are in the image
- Automatic segmentation: This type of segmentation does not need any prior knowledge on either the type of the texture or the number of textures in the image.

There are already a large number of supervised (86; 129) and unsupervised (130; 118) texture segmentation algorithms in the literature. While the supervised and unsupervised techniques are very useful in a lot of applications, it is not very useful for our

application, since for both techniques the number of textures present need to be given a priori. The particular application area with which this thesis is concerned is content-based retrieval of art and museum artefact images, where the segmentation is to be performed on several thousand images. It is therefore inefficient to expect the number of textures to be manually provided for all the images. An automatic texture detection and segmentation algorithm is therefore needed to suit this kind of application.

In this thesis, a novel automatic texture segmentation, i.e. the ones without any a priori knowledge on either the type of textures or the number of textures in the image is presented. The method uses a modified discrete wavelet frames (DWF) decomposition to extract important features from an image before a mean shift algorithm is used together with a fuzzy c-means (FCM) clustering to cluster or segment the image into different texture regions. The proposed algorithm has the advantage of high accuracy while maintaining low computational load. We will also show the advantage of using the modified DWF over the standard DWF and the wavelet transform, and demonstrate how using the mean shift together with the FCM helps in speeding up the fuzzy clustering process.

The next section will briefly describes several popular texture segmentation algorithms, and from there the novel automatic texture segmentation algorithm is derived.

6.2 Review of Texture Segmentation Algorithms

There are already a large number of texture segmentation algorithms in the literature. Texture segmentation usually involves the combination of texture feature extraction techniques with a suitable segmentation algorithm. Among the feature extraction techniques used for texture segmentation are the Gabor filters, Markov random fields, Laws' texture features, fractal dimension, Voronoi polygons, and wavelet decomposition. Segmentation methods, on the other hand, are based on split-and-merge, region growing, estimation theory, clustering, relaxation and neural networks. It is impossible to review all of the available texture segmentation algorithms in the literature, but the following will briefly describes some of the popular techniques as well as the techniques which we believe are of interest for this thesis.

6.2.1 Texture Segmentation Techniques

In (86), Paragios and Deriche combine the Gabor filters with Geodesic Active Contour Model to obtain supervised texture segmentation. First the feature space is generated by filtering the image using Gabor filters, and analysing their responses as a multi-component conditional probability density function. The texture segmentation is then obtained by minimising a Geodesic Active Contour Model objective function where

the boundary-based information is expressed via discontinuities on the statistical space associated with the multi-modal textured feature space.

Manjunath and Chellappa (67) experimented with Gauss Markov random field models (GMRF) to obtain an unsupervised texture segmentation algorithm. The image is first divided into a number of non-overlapping regions and the GMRF parameters are computed from each of these regions. A nearest neighbour clustering method is used to merge these regions. The parameters of the model estimated from the clustered segments are then used in two different schemes, one being an approximation to the maximum a posteriori estimate of the labels and the other minimizing the percentage of misclassification error.

Voronoi polygons are used by Tuceryan and Jain (131) together with a relaxation algorithm in their unsupervised texture segmentation algorithm. The algorithm first builds the Voronoi tessellation of the tokens that make up the textured image. It then computes a feature vector for each Voronoi polygon. These feature vectors are used in a probabilistic relaxation labelling on the tokens, to identify the interior and the border regions of the textures.

Fractal dimension (FD) has also been used for texture segmentation. Chaudhuri and Sarkar (73) use it with a k-means like clustering approach to obtain an unsupervised segmentation. Six FD features are based on the original image, the above average/high gray level image, the below average/low gray level image, the horizontally smoothed image, the vertically smoothed image, and the multi-fractal dimension of order two. A modified box-counting approach is used to estimate the FD, in combination with feature smoothing in order to reduce spurious regions. To segment a scene into the desired number of classes, an unsupervised k-means like clustering is used.

6.2.2 Multiresolution Segmentation Techniques

Multiresolution segmentation is becoming more popular due to the extra information available through different resolutions of the image to be segmented. It performs the segmentation algorithm over a range of spatial scales of the input image (132). Wavelets and other multiresolution algorithms are very suitable for use with this segmentation algorithm. To illustrate multiresolution segmentation, consider the pyramid-structured wavelet transform output image shown in Figure 6.1.

The PWT image in Figure 6.1(a) can be viewed as a pyramid image (Figure 6.1(b)). From the pyramid, it is clear that there are three different image resolutions forming the PWT output. Now the segmentation process can be applied from the top to the bottom of the pyramid. The four sub-images at the top of the pyramid are used as a four-dimensional data set to be segmented. The crude segmentation results at this level are interpolated and passed to the next resolution. The segmentation at the next

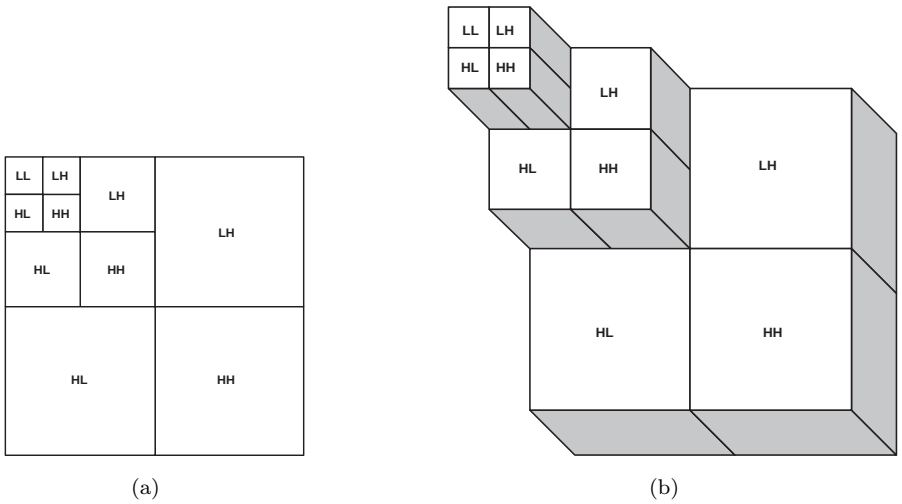


FIGURE 6.1: (a) PWT output image (b) Represented as a pyramid.

resolution can then be performed by combining the data of that resolution with the temporary segmentation obtained from the previous resolution. The process continues and the final segmented image is obtained at the base of the pyramid. From this simple example only, it is clear that multiresolution segmentation offers more advantages over single resolution techniques.

In (133), Salari and Ling use the above multiresolution segmentation algorithm using the pyramidal wavelet transform and k-means clustering. The four channels at the top of the pyramid are grouped into the desired number of clusters by using the k-means clustering technique. The resulting temporary segmentation is then labelled according to different clusters and normalized to avoid the domination of certain channels. The labelled image is then interpolated and combined with the three channels at the next level. The process continues until the labelled image corresponding to the root level is obtained. A post-processing algorithm can be applied to eliminate undesired small segments.

Chang and Kuo (134) experimented with the tree-structured wavelet transform and fuzzy clustering. The image is first decomposed into tree-structured wavelet decomposition. Then, starting from the coarsest level, four leaf nodes corresponding to each tree nodes are clustered using the fuzzy c-means algorithm. The resulting output from the fuzzy clustering is a membership function. This membership function is then interpolated and combined with the leaf nodes or the membership functions available at the next level to provide features at that level. The process continues until the membership function corresponding to the root node is achieved. The segmented image can be obtained by assigning each pixel to the class in which it has the highest membership value.

In (135), Krishnamachari and Chellappa proposed a texture segmentation technique based on multiresolution Gauss Markov random fields (GMRF). Coarser resolution sam-

ple fields are obtained by sub-sampling the sample field at fine resolution. They used two techniques to estimate the GMRF parameters at coarser resolutions from the fine resolution parameters, one by minimising the Kullback-Leibler distance and another based on local conditional distribution invariance. The coarsest resolution data is first segmented by modelling a label field using MRF and the segmentation results are propagated upward to the finer resolution.

6.2.3 Automatic Texture Segmentation

As previously mentioned, there are very few automatic texture segmentation algorithms available in the literature. All of the texture segmentation techniques described above are not automatic texture segmentation algorithms. One of the few automatic texture segmentation technique can be found in (136) by Perry and Lowe. In their algorithm, a modified Gabor transform is used to produce n-dimensional feature vectors for each pixel. To perform segmentation, texture seed regions are established by comparing each feature vector with their neighbours. Elements that are found to be similar to at least three of their four neighbours are placed on the list of candidate texture seed regions. Then texture region borders are extended and refined through an iterative stage. The growing ends when there exists no neighbouring element for which the distance of this element to the region is smaller than the threshold for the region. This algorithm however is computationally very intensive.

Other automatic texture segmentation algorithms mostly tend to first identify the number of textures within the image before an unsupervised clustering algorithm is carried out to segment the image into the desired number of segments. An example for this kind of segmentation can be found in (137). In their paper, Porter and Canagarajah use the standard wavelet transform together with k-means clustering and within cluster distance calculation to perform automatic texture segmentation. The wavelet transform is used to extract texture features, and the within cluster distance calculation is used to estimate the number of different textures within the image. Once the number of textures are known, the k-means clustering is applied to the data where k is the estimated number of texture regions. But this method is rather expensive computationally as it requires two completely different sets of algorithms, one to detect the number of textures present and another to segment them. Furthermore we will show in the next section that the standard wavelet transform features does not provide a particularly good feature space, hence will probably affect the segmentation result for different textures.

6.2.4 Comparison of Texture Segmentation Techniques

There are also a few papers comparing the performance of several segmentation techniques. Du Buf et al. (138) compared seven different texture feature extraction meth-

ods which are the Grey Level Co-occurrence Matrix, fractal, Michelle's texture feature, Knutsson's texture feature, Laws' texture feature, Unser's texture feature, and curvilinear integration. Their paper is one of most important studies since they are the first to attempt to evaluate issues of image segmentation and boundary accuracy comparison in a quantitative framework. From the seven feature extraction methods tested, the Haralick, Laws and Unser methods gave the best overall results.

Chang et al. (139) experimented with three feature extraction methods and three segmentation algorithms. The three texture feature methods are the Grey Level Co-occurrence Matrix (GLCM), Laws' texture feature and Gabor filtering techniques while the segmentation algorithms include the fuzzy clustering, square-error clustering and split-and-merge algorithms. The combination of Gabor filtering with the square error clustering was found to be the best among several combinations. Gabor filtering more readily incorporates multiresolution information than the GLCM and Laws, therefore resulting in much better segmentation.

Pichler et al. (140) compared the pyramidal and tree-structured wavelet transform with the Gabor filtering in segmenting textured images. Fuzzy c-means clustering is used to obtain a segmentation based on computed texture features. The Gabor filtering was found to give the best segmentation result among the three techniques. Nevertheless, Gabor filtering was found to be very time consuming compared to the other two techniques.

6.3 A Novel Automatic Texture Segmentation Algorithm

Our proposed texture segmentation algorithm (141) is based on multiresolution clustering of texture data. Firstly a feature extraction technique is applied to the image to obtain a series of texture coefficients at different resolutions. Each coefficient represents pixels in the original image. The coefficients are then clustered into an appropriate number of groups in the feature space, and each pixel is labelled to the group of its corresponding coefficients. There are several feature extraction techniques to be used in capturing texture coefficients. Since we have been using discrete wavelet frames throughout this thesis, and will eventually use it again to extract textures from the segmented regions following segmentation, it is only logical to use the discrete wavelet frames for segmentation purposes as well.

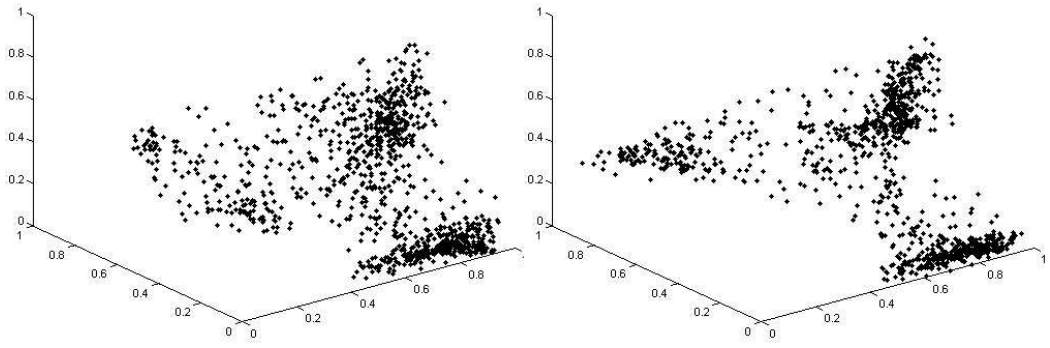
Nevertheless, DWF results in quite a large number of coefficients, and this might slow the segmentation process. A modified DWF is proposed instead. Once the feature space has been constructed, a suitable clustering algorithm can be used to cluster the data. However, since we would like to produce an automatic texture segmentation algorithm, the clustering algorithm needs to be able to identify how many clusters there are in the feature space. This can be done using the mean shift algorithm. The modified

DWF, mean shift and the proposed segmentation algorithms are described in detail in the following.

6.3.1 Modified Discrete Wavelet Frames

Recall that for the standard wavelet transform, the output of the filter is sub-sampled at each level resulting in an output with the same size as the input image, while discrete wavelet frames are an over-complete wavelet transform where all the output data are preserved. For an $M \times M$ image, the output of the standard wavelet transform is M^2 while the output of the discrete wavelet frames will have $(3K + 1) \times M^2$ coefficients, where K is the number of decomposition levels. If we sample the DWF output every 2^k samples, where $k = 1, ..., K$ is the level associated with a particular filtered image, the output will be exactly the same as the wavelet transform. The standard wavelet transform coefficients are actually a subset of a much larger set of discrete wavelet frames coefficients. It is therefore of more computational advantage to use the standard wavelet transform for segmentation instead of the discrete wavelet frames.

However, because of the sub-sampling, the wavelet transform coefficients are of very high variance, which could affect the clustering process quite badly. For example, the 3rd level coefficients of the wavelet transform are sampled every 8 coefficients both horizontally and vertically, and if we plot these into a feature space, there will not be well defined clusters due to the high variance, even after applying some smoothing process. The discrete wavelet frames on the other hand does not suffer from this problem. While the wavelet transform is perfectly reconstructable, thus making it very good in some other fields, it is not very suitable for image segmentation, at least for the automatic case. One of the most important properties in achieving automatic segmentation is to be able to come up with a well defined feature space, thus the wavelet transform is clearly unsuitable. This also contributes to our decision in opting for the DWF instead of PWT in feature extraction techniques in chapter 4.



(a) (b)

FIGURE 6.2: 3D plot in the feature space for (a) wavelet transform coefficient, (b) modified DWF coefficient

Nonetheless, it occurs that we can reduce the coefficients of the discrete wavelet transform to take the pyramid structure of the standard wavelet transform without significantly affecting the distribution of data in the feature space. If the coefficients of the discrete wavelet frames are carefully chosen rather than simply throwing away every other data point as in the wavelet transform, well defined clusters can be preserved and the amount of data can also be reduced greatly. A simple yet reliable method is to take the mean of energy within distinct blocks. For each filtered image, the DWF coefficients are divided into distinct blocks of size $2^k \times 2^k$, and the mean absolute value of the data within the blocks are taken as the new coefficients of the DWF at that level. This results in data reduction of factor $(2^k \times 2^k)$ for that particular filtered image. If this procedure is repeated for every filtered image of the DWF we will have the same pyramid configuration as the wavelet transform but with better coefficients. Figure 6.2 shows a 3D plot of coefficients at level 3 for both the wavelet transform and the modified DWF of the same image consisting of 3 textures. Notice the data of the wavelet transform coefficients are poorly scattered and end up detecting 4 clusters instead of 3.

6.3.2 Mean Shift Algorithm

Mean shift clustering is a relatively new clustering technique which finds possible cluster centres based on the density gradient of data, thus allowing unsupervised clustering to be performed. The rationale behind the density estimation based clustering approach is that the feature space can be regarded as the empirical probability density function $(p.d.f)$ of the represented parameter. Dense regions in the feature space thus correspond to the local maximum of the $p.d.f.$, that is, to the modes of the unknown density. Once the location of the mode is determined, the cluster associated with it can be delineated based on the local structure of the feature space.

Most of the work on mean shift clustering in image processing is done by Comaniciu and Meer (142; 143; 144), although the original idea was introduced by Fukunaga and Hostetler (145). The idea of mean shift is to shift all points in the feature space by a significant amount until they converge to certain points. The convergence points are subsequently analysed to find possible cluster centres. A point is shifted to a new location based on the mean of all points within a radius h from the point itself. Theoretically the point will be shifted towards a local density maximum of the data set. An example of mean shift convergence of a point is shown in Figure 6.3, for a two-dimensional case.

However, Comaniciu and Meer used a simple nearest neighbour clustering to associate each data point to its cluster centre for their colour features. We find this approach is too basic to be used for texture features, since unlike colour, the distribution of texture feature data in the feature space is more complex and therefore needs a more robust clustering technique. Furthermore, since our method uses a multiresolution feature extraction in discrete wavelet frames, the decision about cluster membership for a pixel

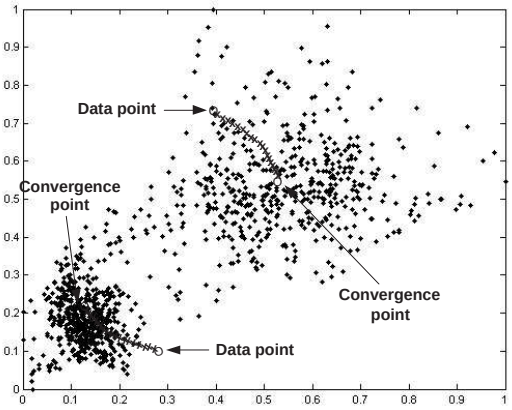


FIGURE 6.3: Mean shift convergence of a point.

only needs to be decided at the root level. The clustering output at all levels, except the root, only serve as intermediate results, and therefore is better represented by some sort of a membership function instead of a membership class. For these reasons, we opt to use the fuzzy c-means clustering to cluster the data, while the mean shift algorithm is just used to estimate the number of clusters and the cluster centres.

The mean shift algorithm used in the proposed segmentation technique can be broken down into five processes:

- **Data sampling.** To reduce the computational load, a set of m points called the sample set is randomly selected from the data. Two constraints are imposed on the points retained in the sample set. The distance between any two neighbours should not be smaller than h, the radius of the sphere $S_h(x)$, and the sample points should not lie in sparsely populated regions. The latter condition is to avoid low density clusters. A region is sparsely populated whenever the number of points inside the sphere is below a threshold T.

- **Mode seeking.** For each of the sample points, apply a mean shift procedure until the points converge to a stationary point. The mean shift computation for each sample points is based on the entire data set. The convergence points are considered as cluster centre candidates.

- **Cluster centres derivation.** Any subset of cluster centre candidates which are sufficiently close to each other (for any given point in the subset, there is at least another point in the subset such that their distance is less than h) defines a cluster centre. The cluster centre is the mean of the cluster centre candidates in the subset.

- **Cluster centre validation.** Between any two cluster centres, a significant valley should occur in the underlying density. The existence of the valley is tested for each pair of cluster centres. If the density at any point between the two centres is

below $V\times$ *(highest density between the two centres)*; $0 < V < 1$, then a valley is observed and both centres are valid. If no valley was found, the cluster centre of lower density is removed from the set of cluster centres.

- **Clusters delineation.** At this stage, each data point is associated with a cluster centre using fuzzy c-means clustering technique.

6.3.3 Segmentation Algorithm

Our proposed texture segmentation algorithm has a hierarchical structure and consists of two phases: a top-down decomposition phase followed by a bottom-up segmentation phase. Figure 6.4 shows the flowchart of the algorithm.

6.3.3.1 Top-Down Decomposition Phase

In the top-down decomposition phase, we perform a K-level discrete wavelet frames decomposition. For a $2^n \times 2^n$ image, this results in $3K + 1$ planes of $2^n \times 2^n$ data. The amount of data is then reduced by applying steps described in the modified DWF section. This results in pyramid-structured coefficients. At this point, we label the original image of size $2^n \times 2^n$ with level index 0 (the root level), the four sub-images of size $2^{n-1} \times 2^{n-1}$ with level index 1 and so on. Since the discrete wavelet frames provide good spatial and frequency energy localization, we may take the energy value of each modified DWF coefficient as an energy feature. However, the variance of the feature is still high since only one sample is used. By assuming that neighbouring DWF coefficients are identically and independently distributed, the variance can be reduced by performing a local averaging or smoothing operation.

On the one hand, it is desirable to have a large window to reduce the statistical variations. On the other hand, since a large window centred at points in the texture boundary region may contain multiple texture classes, the window size has to be small. To avoid this problem, we choose to use a sophisticated adaptive smoothing algorithm developed by Chang et al. (119), which repeatedly implements a simple local averaging operation until some criterion is satisfied. A typical smoothing operator is of the form:

$$W = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{6.1}$$

For an $N \times N$ image $f(x, y)$, the iteration stopping criterion for a sub-image at the pth

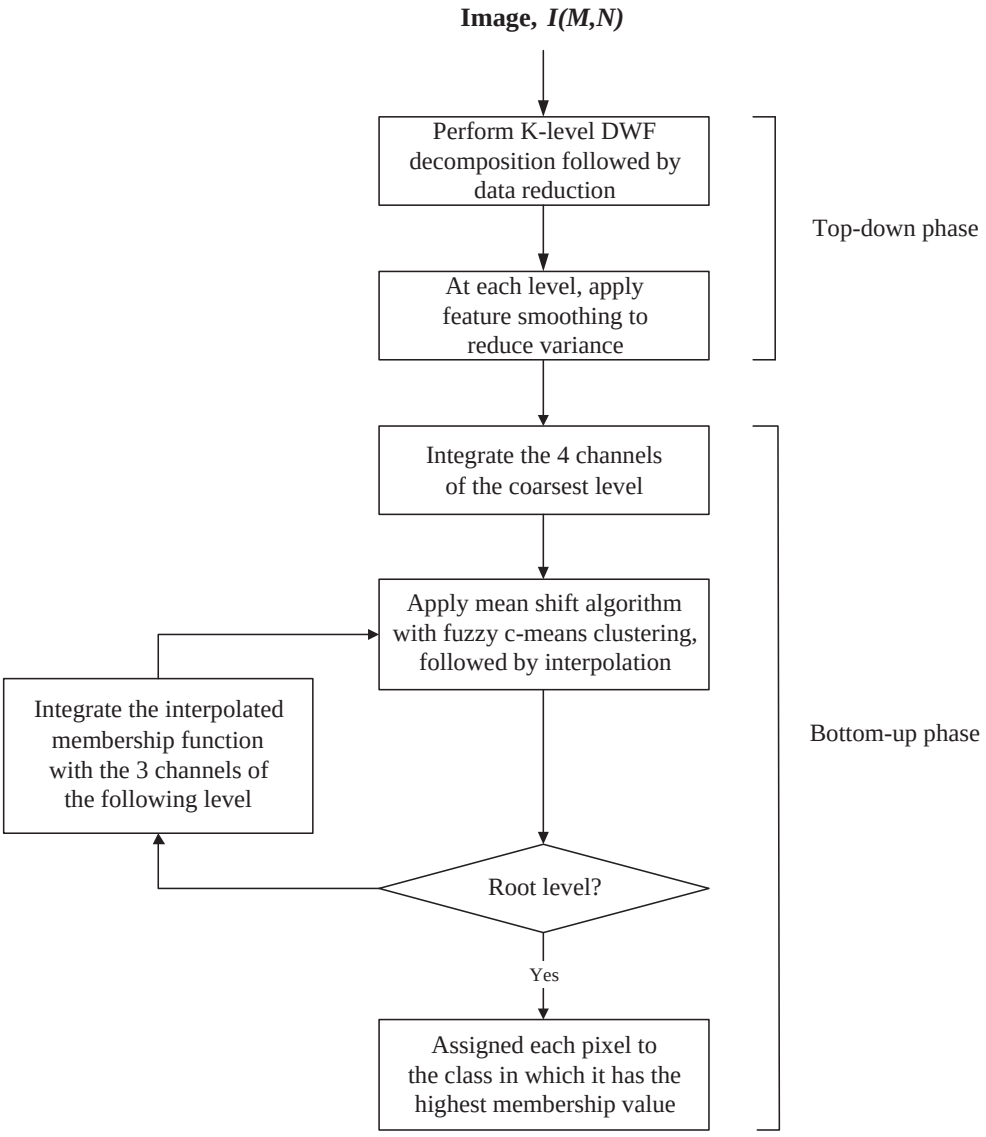


FIGURE 6.4: Flowchart of the proposed segmentation algorithm.

level is given by:

$$\Gamma_k = 1.28 \frac{\alpha}{2^p} N^2 \tag{6.2}$$

where

$$\Gamma_k = \sum_x \sum_y \frac{\left| D^k f(x,y) \right|}{|W^{k-1} f(x,y)| + |D^k f(x,y)| + \epsilon} \tag{6.3}$$

and α is an estimate of the percentage of the number of boundary pixels (suggested value 1/N), ϵ is a very small number, $D^k \equiv W^k - W^{k-1}$, and W^k means applying the smoothing operator k times.

The smoothed energy values are then normalized to the range between 0 and 1 within each node so that they can be conveniently used for segmentation in the bottom-up phase, as well as to make sure that no components will artificially dominate the clustering process.

6.3.3.2 Bottom-Up Segmentation Phase

In the bottom-up phase, we start with level K and produce an intermediate segmentation result for level $K-1$ using the four sub-images available at level K. To generate the intermediate segmentation, the four sub-images of size $2^{n-K} \times 2^{n-K}$ are integrated in a way that it can be viewed as four-dimensional $2^{n-K} \times 2^{n-K}$ data, before the mean shift algorithm is applied and provides us with the number of clusters detected in the data as well as the cluster centre positions. As mentioned in the last section, the fuzzy c-means clustering (FCM) is chosen above other clustering techniques and is applied to the four-dimensional data using the information provided by the mean shift. The fuzzy c-means clustering algorithm is an iterative procedure described in the following:

Fuzzy C-Means Clustering Algorithm

Given M input data points $\{x_m; m = 1, ..., M\}$, the number of clusters C $(2 \leq C < M)$, and the fuzzy weighting exponent w, $1 < w < \infty$, initialize the fuzzy membership functions $u_{c,m}^{(0)}$ with $c = 1, ..., C$ and $m = 1, ..., M$ which are the entry of a $C \times M$ matrix $U^{(0)}$. Perform the following for iteration $l = 1, 2, ...$:

1. Calculate the fuzzy cluster centres ${v_c}^l$ with $v_c = \sum_{m=1}^{M} (u_{c,m})^w x_m / \sum_{m=1}^{M} (u_{c,m})^w$
2. Update $U^{(l)}$ with $u_{c,m} = 1/\sum_{i=1}^{C} \left(\frac{d_{c,m}}{d_{i,m}}\right)^{\frac{2}{w-1}}$ where $(d_{i,m})^2 = \|x_m - v_i\|^2$ and $\|\cdot\|$ is any inner product induced norm.
3. Compare $U^{(l)}$ with $U^{(l+1)}$ in a convenient matrix norm. If $\|U^{(l+1)} - U^{(l)}\| \leq \varepsilon$ stop; otherwise return to step 1.

The value of the weighting exponent, w determines the fuzzyness of the clustering decision. A smaller value of w, i.e. w is close to unity, will give the zero/one hard decision membership function, and a larger w corresponds to a fuzzier output. Our experimental results suggest that $w = 2$ is a good choice. The advantage of using the mean shift algorithm together with the fuzzy c-means clustering is demonstrated here. The fuzzy c-means algorithm is not a fully unsupervised clustering method as it requires the number of clusters to be known a priori. Besides that, one other drawback of the fuzzy c-means is finding the best way to initialize the fuzzy membership function.

The FCM algorithm finds a local minimum of $\Sigma_{c=1}^{C}\Sigma_{m=1}^{M}u_{c,m}^{w}d_{c,m}^{2}$ by solving $u_{c,m}$ and v_c, and its output depends on the initial value of $U^{(0)}$. Various methods have been proposed on the best way to initialize $U^{(0)}$ such as the maximin distance algorithm. But by using the mean shift, it not only provides the FCM with the number of clusters, but also the cluster centres, meaning that the initial value of $U^{(0)}$ is already quite close to the final value of $U^{(0)}$. Hence part of the task of the FCM, that is to find appropriate cluster centres, is done. Experiments have shown that the FCM algorithm terminates after just a few iterations, thanks to the precise location of the cluster centres.

The output of the fuzzy c-means is a $2^{n-K}\times 2^{n-K}$ membership function of N_c dimension, where N_c is the number of clusters. Each element of the membership function describes the membership value with respect to a particular type of cluster and the sum of these elements is equal to 1. The membership function is then interpolated to size $2^{n-K+1}\times 2^{n-K+1}$ so that it has the same size with the data at the following level. For simplicity, a linear interpolation algorithm is used to interpolate the membership function.

At level $K-1$, the interpolated membership function is integrated with the 3 sub-images at this level resulting in an (N_c+3)-dimensional data to be used for the next mean shift and FCM processes. These procedures of data integration, mean shift, clustering and interpolation are applied recursively from bottom to top so that we eventually obtain the segmentation result of the root level, i.e. the original image. The final crispy segmentation at level 0 can be determined by assigning each pixel to the class where it has the highest probability of membership. Note that the number of clusters detected by the mean shift algorithm can be different at each level. We might get a wrong number of clusters in the bottom level, but that is just an intermediate result, where not all data are utilized. What matters is the final segmentation result, after all data is taken into account. The incorrect number of clusters in the bottom level might be refined by the data at the higher levels.

Finally the proposed texture segmentation algorithm works on image of any resolution, and not confined to dyadic length image only. For non-dyadic image length, the formula for the modified DWF remains the same except we introduce the $floor$ function when converting the coefficient to lower level. For example, an image with length 355 will have the length of 177 for level 0 coefficients, 88 for level 1 coefficients, and 44 for level 2 coefficients. During the bottom-up segmentation process, the length will be converted back to the original length.

6.4 Experimental Analysis

We will first show the sequence of segmentation result from coarse to fine resolution in order to give a better illustration of the segmentation process at each level. For illustration purpose, we experiment with a composite texture image comprising 4 different

Brodatz textures (D017, D024, D055 and D077), with each texture positioned at each quarter of the image. The size of the image is 256×256. The image is decomposed using the modified DWF for up to three levels using an 8-tap *Daubechies* wavelet filter. The radius, h and threshold, T for the mean shift algorithm is critical, and from experiment a suitable value of h at all levels is found to be 0.2, while a suitable value of T is one twentieth of the total data points at each level. The valley threshold, V is set to 0.5. Figure 6.5 shows the sequence of segmentation results obtained at each level for the composite texture image. In this example, the initial segmentation result obtained at level 2 already gives a quite good segmentation and is used as a basis for higher level processing. It can be clearly seen that as the level increases, the segmentation result improves. This implies that the coefficients from the higher levels help in refining the boundary of the textures, thus illustrating the advantage of multiresolution segmentation.

We will now evaluate the performance of the algorithm for different numbers of textures within an image. The following section evaluates the performance on composite textures, synthetic textures, real scene images and museum images.

6.4.1 Composite Texture Images

In this section, the performance of the segmentation algorithm will be evaluated by its ability in identifying the correct number of textures in the image, as well as its precision in defining the boundaries of the segmented images. The precision is measured by computing the percentage of misclassified pixels in the segmented images. We applied our texture segmentation algorithm to several images of composite textures with size 256×256 pixels and 256 grey levels. Textures from the Brodatz album are used to make up the composite texture images by cut-and-paste technique. Textures pasted are of either rectangular or square shape in order to make the computation of misclassified pixels easier. None of the textures used in our experiment can be discriminated by grey level values alone.

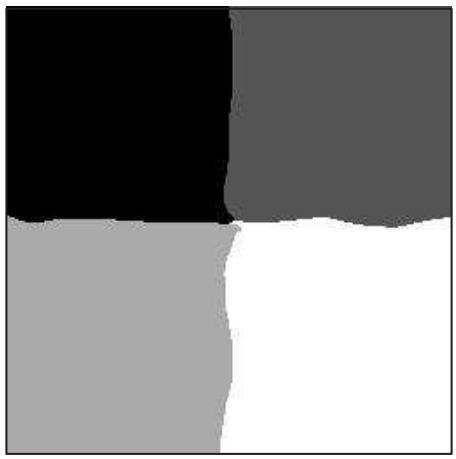
Figure 6.6 shows an example of applying the texture segmentation algorithm to a number of images with different numbers of textures. The 2-textured image consists of texture D012 and D017, the 3-textured image of texture D054, D074 and D102, the 4-textured image of texture D001, D011, D018 and D026, while the 5-textured image consists of texture D001, D053, D065, D074 and D102. All the results in Figure 6.6 show a correctly identified number of texture as well as good segmentation. All together, we have applied our algorithm to 50 composite textures, and the results are summarized in Table 6.1.

A return of 90% correctly detected number of textures is very promising. Except for one of the 3-textured images, which the algorithm detected to have 5 textures, all other incorrect results only miss by plus/minus one texture. Figure 6.7 shows an example of a wrongly detected number of textures. The image consists of texture D065, D066, D086

(a)

(b)

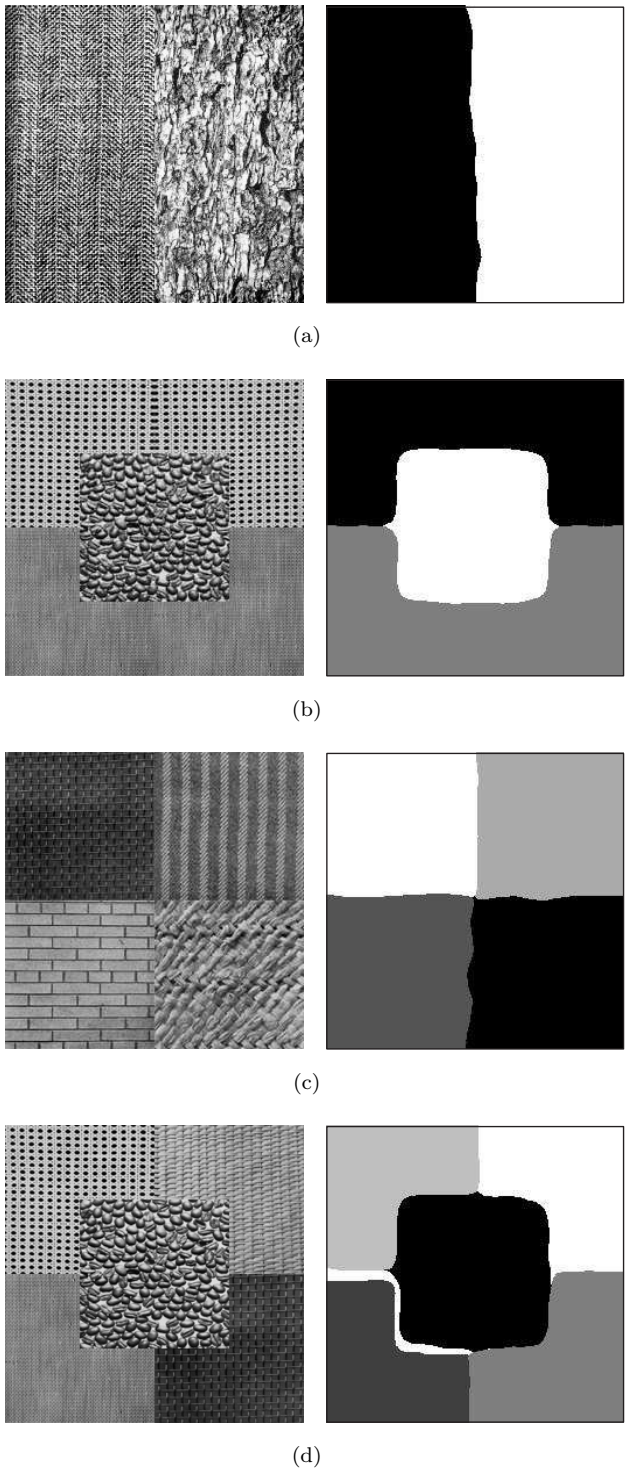
(c)



(d)

FIGURE 6.5: (a) 4-textured image, and its segmentation result at, (b) level 2 (64×64), (c) level 1 (128×128), (d) level 0 (256×256, final result)

and D102. From the figure, it is clear that the incorrect segmentation is caused by the fact that the top half texture appears to contain two visually different regions. For the 5 incorrect cases, the cause is either the same problem as above, or the fact that two textures are almost visually the same.



(a) (b) (c) (d)

FIGURE 6.6: Segmentation result for different number of textures

Table 6.2 shows the percentage of segmentation errors for the five correctly segmented textures shown in Figure 6.5 and 6.6. All of the images give an error percentage of below 5% which is quite a low rate in texture segmentation. Notice that the more texture boundaries there are, the more difficult decisions must be made, resulting in an increasing number of misclassified pixels. Non-boundary pixels seem to be well distinguished by

Number of textures in an image	Number of images tested	Images with correctly detected number of textures	Number of textures detected for the wrong detection case			
			2	**3**	**4**	**5**
2	16	15		1		
3	12	11				1
4	13	12				1
5	9	7			2	
Total	50	45 (90%)		1	2	2

TABLE 6.1: Percentage of correctly detected number of textures.

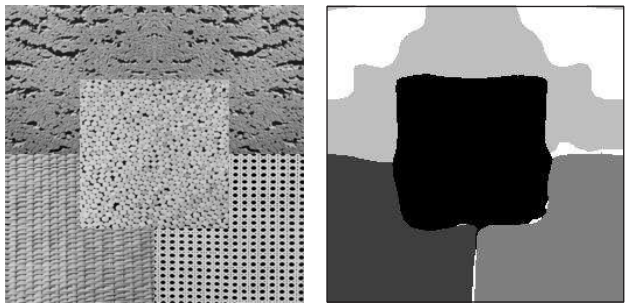


FIGURE 6.7: Example of incorrect segmentation

the proposed algorithm. All together from the 45 correctly segmented images, we obtain an average of just 3.72% misclassified pixels.

Finally, we compare the performance of our algorithm with a segmentation technique based on the wavelet transform segmentation, i.e. the data to be clustered by the mean shift and the FCM is generated by the wavelet transform instead of the modified DWF. Figure 6.8 compares the performance of the modified DWF with the wavelet transform method for the image in Figure 6.6(d). Clearly, the wavelet transform fails to provide good clusters in the feature space resulting in a poor segmentation. Also, the sampling of poorly scattered data points during the mean shift results in a rather inconsistent segmentation for the wavelet transform-based segmentation. The modified DWF is therefore superior to the wavelet transform in terms of segmentation performance, and is superior to the standard DWF in terms of computational speed.

Image	**Textures**	**Misclassified pixels**	**Percentage of error**
Figure 6.5	4	1654	2.52%
Figure 6.6(a)	2	621	0.90%
Figure 6.6(b)	3	1771	2.70%
Figure 6.6(c)	4	1615	2.46%
Figure 6.6(d)	5	2816	4.29%

TABLE 6.2: Percentage of misclassified pixels

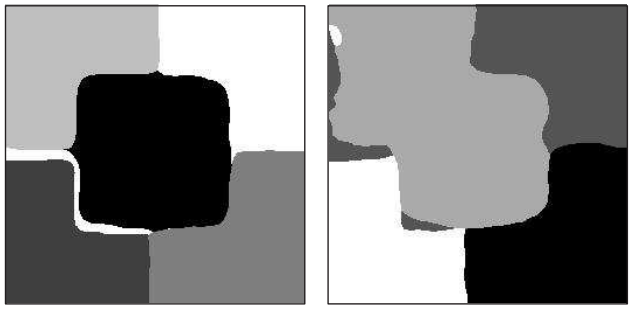


FIGURE 6.8: Result using modified DWF (left) and wavelet transform (right)

6.4.2 Synthetic Texture Images

Figure 6.9 shows segmentation results for synthetic textures composed of the +- and L- symbols. This texture pair has a spectrum with the same magnitude but different phases. However since it is difficult to pre-assign the classes for boundary pixels, it is difficult to compute the misclassified pixels. Thus the evaluation for this particular problem is simply based on visual inspection. The algorithm successfully segments the two different textured regions. Since the illumination of both textures is the same, this example also shows that it is the surface texture, not the illumination condition that is being classified.

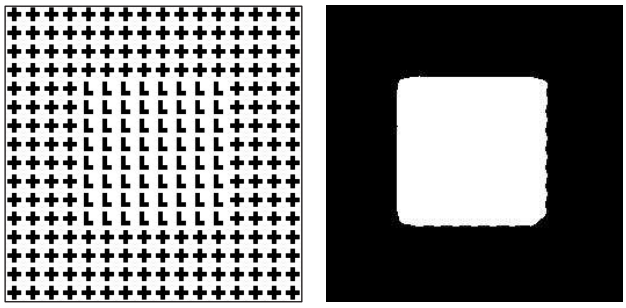


FIGURE 6.9: Segmentation result of synthetic textures

6.4.3 Real Scene Images

Figure 6.10 shows a segmentation result on a real scene image, where the algorithm correctly segmented the sky, mountain and water into 3 separate regions. As in the synthetic texture case, it is difficult to define an objective boundary for this example, thus the percentage of segmentation errors cannot be measured. However, from the figure, it is clear that the proposed algorithm works well in distinguishing real scene image textures.

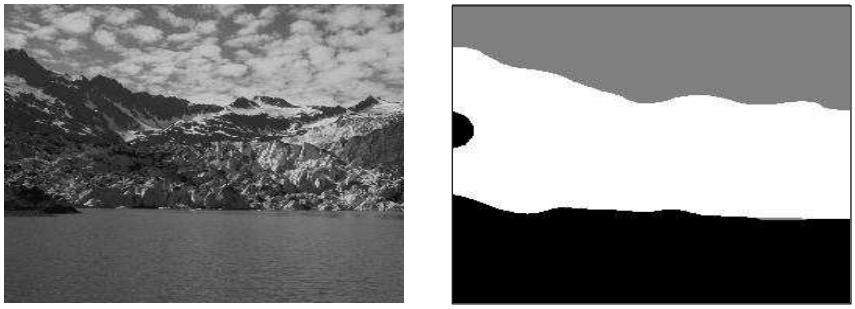


FIGURE 6.10: Segmentation result of real scene image

6.4.4 Museum Images

Figure 6.11 shows an example of one of many various kinds of images in our museum collections. The top figure shows an image of an open book consists of a textured region in the middle of the book and non-textured region in other part of the book as well as non-textured background. The bottom left figure shows the segmentation result which consists of 3 segments, one of which is textured. From the result of the segmentation, it can be seen that the algorithm manages to isolate the textured region of the book page from the non-textured region. The bottom right figure shows the particular textured region. The result suggests that the proposed segmentation algorithm will be useful in texture retrieval application.

FIGURE 6.11: Segmentation result of museum image

6.4.5 Computational Speed of The Algorithm

The computational speed of the proposed algorithm depends on the resolution of the image as well as how many segments the algorithm perceives the image has. The higher

the number of different clusters, the longer it will take to segment the image. Segmentation on typical museum image of size 768×768 with around 3 to 5 segments take around 10 seconds on average. This will be very helpful in minimising the time spent when creating a database of very large image collections.

6.5 The Effect of Segmentation Parameters

Throughout the entire segmentation process, several parameters are encountered and these parameters may or may not have significant effect on the outcome of the segmentation result. These parameters include the radius, h, threshold, T and valley, V of the mean shift algorithm, the fuzzy weighting exponent, w and the fuzzy stopping criterion, ε of the fuzzy clustering, and the estimate of the percentage of the number of boundary pixels, α in the adaptive smoothing algorithm. In this section we will discuss the significance of these parameters to the final segmentation result.

The mean shift algorithm is used to determine the number of texture regions in our segmentation algorithm, hence the three mean shift parameters affected only the final number of texture segments. The fuzzy clustering and adaptive smoothing parameters on the other hand contributed to the boundary accuracy of the segmentation of the known number of texture regions.

6.5.1 Mean Shift Parameters

The radius and threshold of the mean shift is very crucial in our segmentation algorithm. The radius controls the number of segments or clusters in the image. It needs to be big enough so that the mean shift can converge to the correct cluster centers, but cannot be too big as it will result in all the points converging to the same cluster center, i.e. only one big cluster is found. The threshold is used to make sure that the sample points do not belong to scarcely populated area. This in turn contributes in controlling the size of the texture segments, i.e. in order for a particular texture to be considered as one texture region, the textured area should not be too small.

The radius and threshold is inter-related. When the radius is small, an appropriate threshold needs to be found so that it proportionates with the amount of points in the circle of radius h. Figure 6.12 illustrates the effect of this inter-relation. A very large radius, depending on the threshold, will either detect only 1 big cluster or no cluster at all. Small radius with small threshold will produce too many clusters, thus increasing the probability of error. However since we normalized the wavelet features to be between 0 and 1, the choice of radius and threshold can be determined quite easily by experiment. A radius of 0.2 is found to be suitable for our collection of museum images, while the

value of the threshold, assuming there will not be more than 20 clusters within the image, is taken as one twentieth of the total data points at each level.

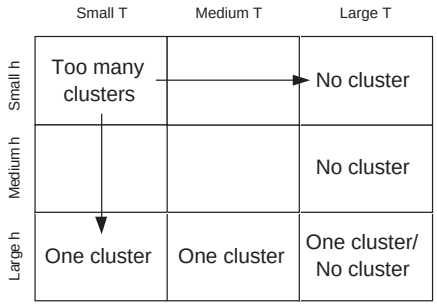


FIGURE 6.12: Inter-relation of radius, h and threshold, T

The last mean shift parameter, the valley, V is less crucial and is only used to check whether two cluster centers are actually in two different clusters. 100 points are generated between the two cluster centers and the density of each point within the circle radius is calculated. If at any point the density is lower than $V \times$*(highest density of the two cluster centers)*, then it proves the two cluster centers are in different clusters. Otherwise the two are considered to be in one cluster, and only one of the cluster centres will be considered as the final cluster center. Therefore the value of V does effect the number of clusters; the bigger V, the higher the number of clusters. It was found that a suitable value of V is 0.5.

6.5.2 Fuzzy Clustering and Adaptive Smoothing Parameters

As mentioned before, the fuzzy weighting exponent, w $(1 < w < \infty)$ controls the fuzziness of the membership function of the fuzzy clustering algorithm. The higher the weighting exponent, the more accurate the boundary accuracy of the segmentation. Nonetheless, bigger w also implies that there will be 'holes' within the homogeneous texture segments. In other words, while it may increase the boundary accuracy of the segments, the non-boundary region will not be as smooth. This problem can be solved by using higher w together with some relaxation process. However as bigger w tend to make the membership function to be uniform, it does have an effect during the mean shift convergence of the following level. For this reason the choice of w value should not be too big. From experiment, $w = 2$ was found to be the most suitable choice.

The fuzzy stopping criterion, ε does not have a major impact on the segmentation result since it only effects the location of the final cluster centers. The smaller the stopping criterion, the more iteration the fuzzy clustering will perform before settling the location of the cluster centers. However there is very little difference on the cluster center location that overall this parameter does not have a significant impact on the final outcome.

Finally, the estimate of the percentage of the number of boundary pixels, α controls the

shape of the clusters. The adaptive smoothing operation is applied to reduce the high variance of the wavelet coefficients, and the number of times the smoothing iteration is applied depends on α. Smaller α means applying the smoothing operation more times and may results in the data points being grouped together in one big cluster. Higher α, the extreme case means no smoothing operation at all, on the other hand it may result in loosely populated clusters and might lead to too many clusters. The choice of $\alpha = \frac{1}{N}$ as suggested by (119) is suitable for our application, where N is the length of the image.

6.6 Texture Identifier for CBIR

In order to be used with a content-based image retrieval system, it is necessary for the segmentation algorithm not only to segment between textured region, but also to distinguish between textured region and non-textured region. Hence only the feature vectors of the textured segments will be created and stored for matching purposes. Our proposed segmentation algorithm can discriminate not only between different textured regions but also between textured regions and non-textured regions, although it may or may not be able to segment different homogeneous or non-textured region. For example an image consisting of two textures on a dark background at the top and a bright background at the bottom will results in either 3 or 4 clusters using the proposed algorithm, two of them textured. The inability to segment non-textured region however is not important as we are only interested in the textured region. The non-textured region can be discarded using the algorithm to determine whether a particular segment is textured or not proposed by Porter and Canagarajah (137). The basic idea behind their algorithm is to find the ratio of the mean energy in the low frequency channels to the mean energy in the middle frequency channels. Non-textured images (in which the grey level varies smoothly) are heavily dominated by the low-frequency channels in their wavelet transform. However textured images have large energies in both the low and middle frequencies. For a 3-level decomposition, the ratio can be computed between the mean energy in the four low frequency channels (LL_3, LH_3, HL_3, HH_3) and the mean energy in the three middle frequency channels (LH_2, HL_2, HH_2) and is given as follows:

$$R = \frac{LL_3 + LH_3 + HL_3 + HH_3}{LH_2 + HL_2 + HH_2} \tag{6.4}$$

If the ratio,R of a segment is above a certain threshold, then we can conclude that the particular segment is non-textured and thus no feature vectors should be created for it. Otherwise if the ratio is below the threshold, then a feature vector is created for the segment.

From visual inspection, a textured region usually gives a ratio of less then 10, while a non-textured region can be from 10 up to infinity. However this is not always true as our collection of images is very large and it is impossible to visually inspect the ratio

of all the textures. To avoid a situation where a genuine texture is missed, a threshold of 20 is used. It is less harmful for a non-textured being classified as textured than a textured being classified as non-textured as it will be completely ignored in creating feature vectors. Figure 6.13 shows the same image example used in section 6.4.4, and the ratio of each of its segments. The textured segment clearly gives a small ratio to indicate its textured-ness while the two non-textured regions have a much larger ratio.

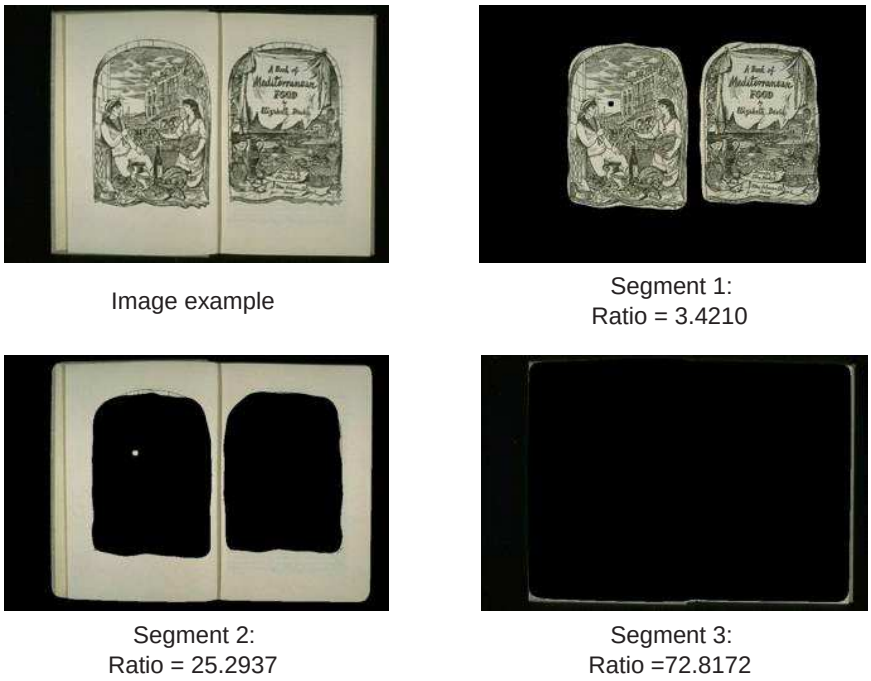


FIGURE 6.13: Example of texture identifier

The texture identifier hence is a useful tool in identifying significant texture regions within an image for feature extraction, while neglecting the non-textured region of the image.

6.7 Integration With A Retrieval System

We are now ready to integrate the proposed segmentation algorithm with a retrieval system. From the segmented regions, the last thing we need to do is to compute the feature vectors for the identified texture regions. We do not need to perform the wavelet decomposition again as we already have the original coefficients of the DWF when we perform the segmentation. In order to make a fair comparison with the multiscale-based retrieval later, all the parameters used for feature extraction and comparison are set to be as similar as possible to the ones used in chapter 4 (table 4.13).

The number of decomposition levels and the wavelet basis are already similar since we use 3 level of decomposition using Daubechies 8-tap wavelet during the segmentation process. The mean subtraction can be applied by removing the local mean of the segmented region

in the LL channel to make it zero mean. The normalized Euclidean distance is used as the distance metric, while the standard deviation energy and the number of zero-crossings of all channels are used as texture features. For the identified texture regions, the features are computed from the original DWF coefficients (not the modified ones used for segmentation) within the segmented texture regions only. This will create a feature vector that closely resembles the texture in that particular region. Finally the luminence function is used to convert colour images to monochrome images. The only dissimilarity here between the multiscale-based approach and the segmentation-based approach is the type of image padding used. Since the same DWF coefficients are used to segment the image as well as to compute the features, the resulting features correspond to symmetric padding as oppose to periodic padding for the multiscale-based approach. Therefore in the segmentation-based approach, the translation invariance property is lost. Nonetheless, since it is impractical to have to compute the DWF again just to preserve the one property, the small dissimilarity can be neglected as it is not going to have much effect in the bigger context of retrieval performance.

One might argue that we can simply take the cluster centers in each level and combine them to provide the feature vector. However this is not true since the number of clusters found in each level might not be the same. Hence it is difficult to compute the feature vector this way. Moreover the cluster center is based on the modified DWF, and not from the true DWF coefficients, thus might not be as good as the original DWF coefficients when used in texture matching between several thousands texture images. Therefore the proposed feature computation above will be used instead, as it is much easier to perform, and the discrimination performance of the original DWF coefficients has been proved in chapter 4.

A simple retrieval experiment is carried in order to observe the validity of this retrieval approach as well as its performance. The retrieval experiment is tested on the same museum database of 1106 images used in the multiscale experiment in the previous chapter. As in the multiscale experiment, the retrieval performance on the museum database was observed visually since we do not know how many similar textures there might be in the database. Nonetheless, as long as the top matches are visually similar to the query, it can be considered successful. Figure 6.14 shows three examples of the segmentation-based retrieval. The yellow line circling part of the image indicates the segmented region found by the algorithm to be similar to the query.

As can be seen, the retrieval system manages to retrieve visually similar textures to the query, and thus is very useful in texture retrieval of museum collections. However, one disadvantage of using segmentation-based retrieval can be seen in the third example. Here the query image is a small stripe from one of the images in the database, and can be considered as a subset of a coarser texture. However since the segmentation algorithm segments the coarser texture, the retrieved images does not actually correspond to the finer texture of the image; instead it corresponds to the coarser texture. In this example,

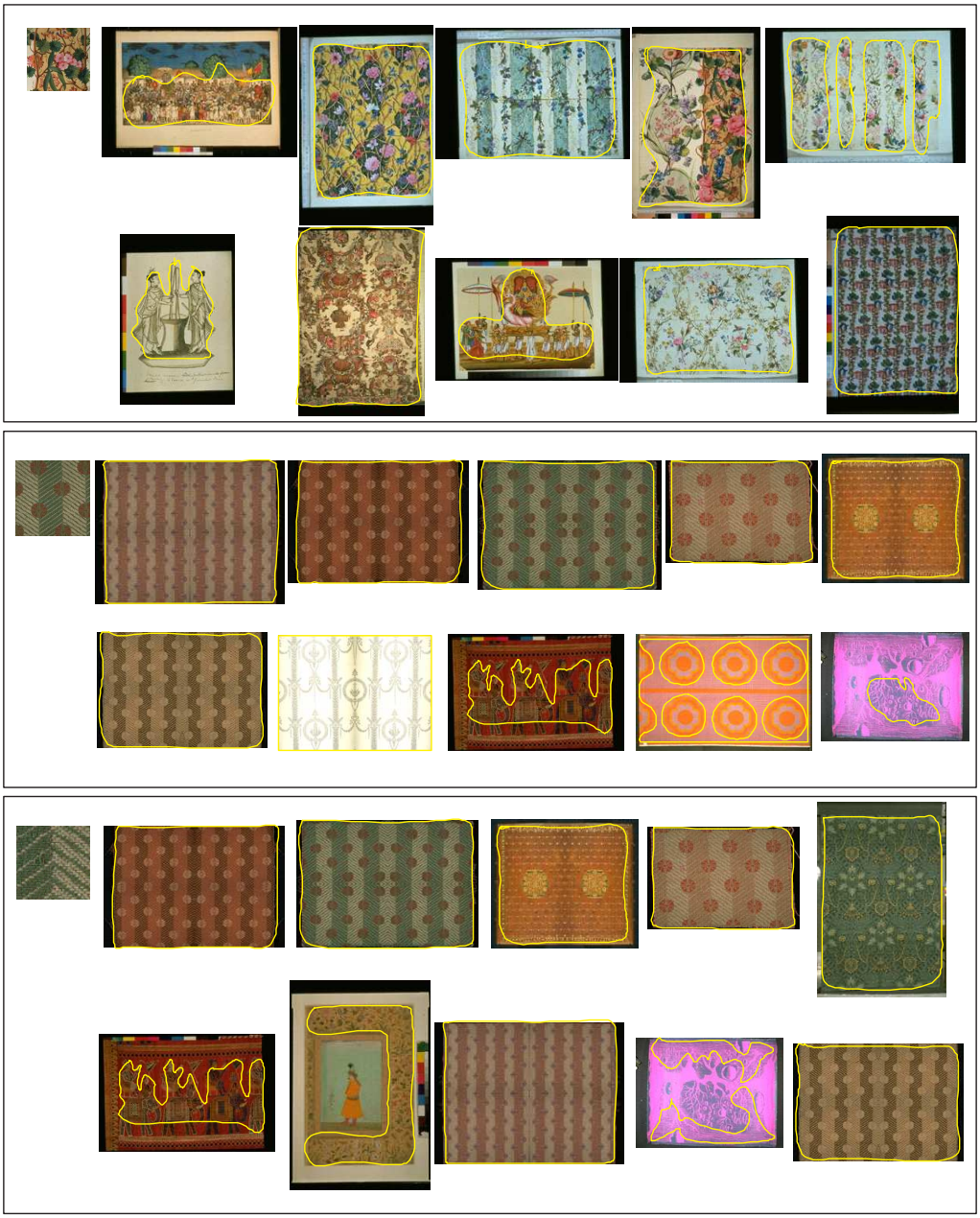


FIGURE 6.14: Example of retrieval results of real museum collections

the features of the finer scale texture and the coarser scale texture may be close (because the majority of the image consists of stripes), hence it still manage to retrieve all images consisting stripes although its accuracy is not as good as the performance using the multiscale-based retrieval (Figure 5.22).

Nonetheless, apart from the lack of multiscale property, the segmentation-based approach proves to be very good and provides an alternative way of texture retrieval to the multiscale-based approach, and it will be interesting to compare the performance of the two. The comparison will be discussed in detail in the next chapter.

6.8 Chapter Summary

In this chapter, a new framework for automatic texture segmentation based on modified discrete wavelet frames and the mean shift algorithm is developed. By modifying the discrete wavelet frames, much better clustering is obtained on a reduced set of data, making possible the use of the mean shift algorithm to detect the correct number of clusters, and substantially reduces the processing time. The mean shift also provides the position of the cluster centres which effectively solves the problem of initializing the membership function in the fuzzy c-means algorithm, and hence reducing the fuzzy iterations.

From the results of the experiments we can see that the proposed method can detect the correct number of clusters as well as segmenting the image correctly in composite textures, synthetic textures, real scene images and museum images, while maintaining the low computational load. The texture segmentation was then extended to be a texture identifier in order to use it in image retrieval system. This is done by computing the ratio of energy in the low-frequency channels to the energy in the middle frequency channels, and observing whether the ratio is below a certain threshold. The feature vector of the identified texture region is then computed for matching purposes. From experiment, the proposed segmentation-based retrieval system performs well in retrieving similar texture from a museum database, hence provides an alternative to the multiscale-based texture retrieval.

Chapter 7

Content-Based Image Retrieval of Museum Images

This chapter introduces the various image collections available on the server of the School of Electronics and Computer Science of the University of Southampton. The three algorithms are then evaluated on these different museum databases of different sizes and observations are discussed. Finally the integration of the proposed algorithms on the Artiste and Sculpteur projects are presented.

7.1 Museum Databases

The School of Electronics and Computer Science of the University of Southampton has collaborations with several museums across Europe through the two major projects carried by the Intelligence, Agents and Multimedia (IAM) Research Group, the Artiste and Sculpteur projects. This program of research is concerned with media processing, extraction of semantics and architectural developments to provide more effective multimedia systems using content and concept based browsing, retrieval and navigation techniques. Four major European galleries are involved in the projects, namely the National Gallery and the Victoria and Albert Museum in London, the Research and Restoration Centre for the Museum of France (C2RMF) in Paris and the Uffizi Gallery in Florence. All of these museums contributed their images to the IAM research group for various research purposes. The Victoria and Albert Museum for instance contributed tens of thousands of its images.

Throughout this final experiment, we will use the images from the National Gallery, the Victoria and Albert Museum and the C2RMF databases. The Uffizi Gallery however will not be used since the number of images in its databases is too small to make useful observations. The three databases to be used are described as below.

7.1.1 The National Gallery

The National Gallery in London houses one of the greatest collections of European painting in the world. Their permanent collection spans the period from about 1250 to 1900 and consists of Western European paintings. Artworks from various well known artists such as Vincent van Gogh, Claude-Oscar Monet and Leonardo da Vinci can be found in the gallery. To facilitate a digital image library within the museum, these paintings are scanned using a very high resolution scanner.

There are more than 1000 images from the museum available in our server database. The original images from this museum are of very high quality and their dimension can be up to 3000 $\times$ 3000 pixels. However scaled-down versions of the images are also available, and in order to compensate for the lack of multidimensional indexing techniques for the proposed algorithms, one of the scaled-down versions will be used for content-based image retrieval experiments. The images of this scaled version have the resolution between 500 to 1000 pixels length. In total, the number of images in the National Gallery database to be used in the final experiment is 1462. Figure 7.1 shows some example of the images within the National Gallery database.

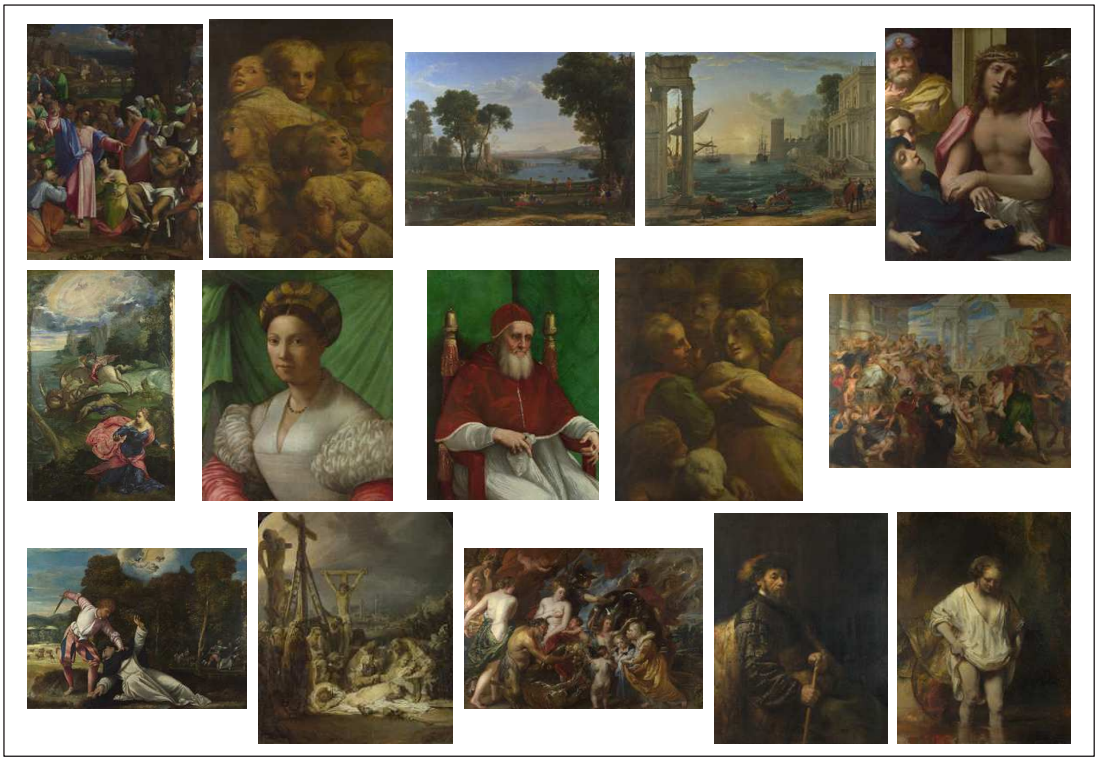


FIGURE 7.1: Example of National Gallery images

7.1.2 The Victoria and Albert Museum

The Victoria and Albert Museum is a great museum of applied and decorative arts. It is a world treasure house with collections of vast scope and diversity. The museum holds 3000 years' worth of artefacts from many of the world's richest cultures. Among its collections are Asian and Islamic arts, Europeans arts and sculptures, twentieth century arts, jewelries, silvers and metals, textiles, dresses, ceramics, glasses, paintings, photography and drawings.

The Victoria and Albert Museum contributed more than 30,000 images to the research group. Except for the paintings and drawings, images from the museum mostly contain pictures of an individual work of art on a homogeneous background. All the images in this collection however are not as high in resolution as the National Gallery images. The image dimension varies between 500 to 1000 pixels length as in the scaled-down version of the National Gallery images. In total, the number of images in the Victoria and Albert Museum database to be used in the final experiment is 16959. Figure 7.2 shows some example of the images within the Victoria and Albert Museum database.

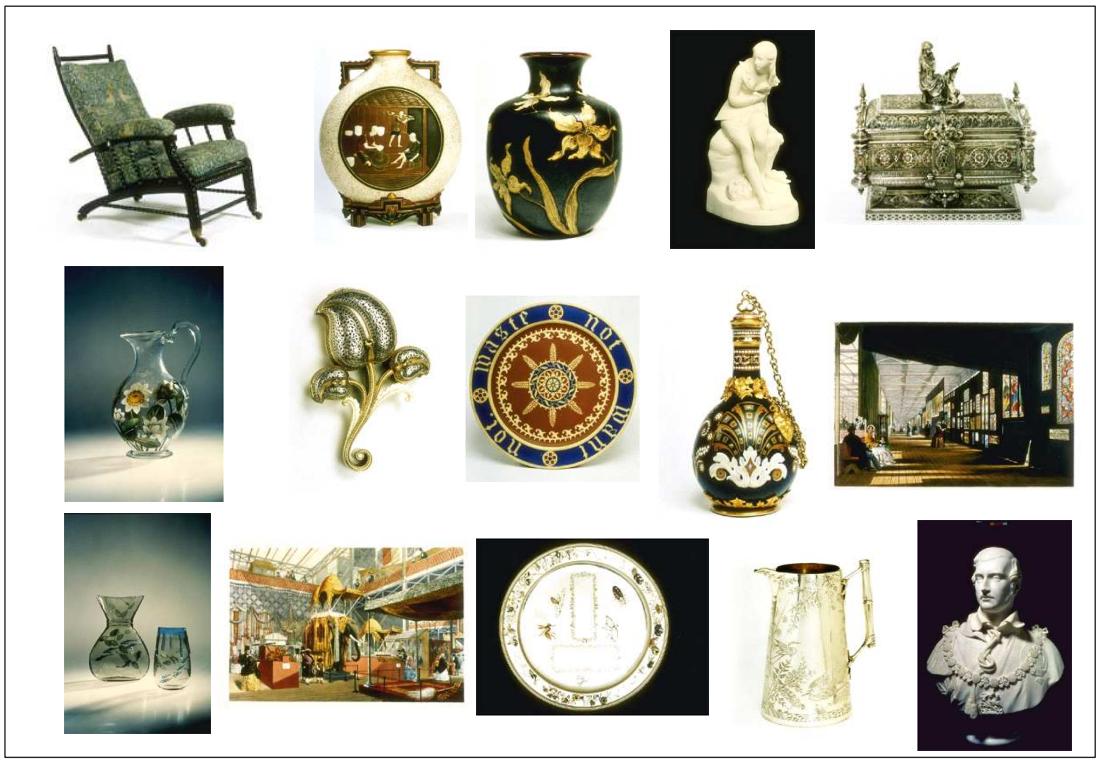


FIGURE 7.2: Example of Victoria and Albert Museum images

7.1.3 The Research and Restoration Centre for the Museum of France (C2RMF)

C2RMF is one of the oldest and largest research laboratories in the museum sector. Its main research area is the restoration of paintings in French museums. C2RMF contributes a large number of images to the IAM research group, in the region of hundreds of thousands. Most of these images are monochrome paintings, and is mainly used for research on image restoration, crack and plank detection and analysis. However, most of these images are duplicates with some processing (restoration) applied. For example, there might be five different versions of the same painting, and on top of it, the image of the plank at the back of the paintings are also included. Since our objective is to perform content-based image retrieval on museum images, the duplicates and the plank images are ignored. Hence a total of 2500 different paintings are selected for use in the CBIR experiments in this thesis. Figure 7.3 shows some example of the images within the C2RMF database.

FIGURE 7.3: Example of C2RMF images

7.2 Content-Based Image Retrieval of Different Museum Databases

The *query by low-quality image* algorithm and the two *query by texture* algorithms are now used for content-based image retrieval of museum collections. The 20 fax images

Query Image No.	Rank of Original	Query Image No.	Rank of Original
1	4	11	2
2	4	12	1
3	1	13	2
4	1	14	4
5	1	15	1
6	6	16	1
7	1	17	1
8	2	18	1
9	1	19	1
10	1	20	1

TABLE 7.1: Retrieval results using 20 fax images on the Victoria and Albert Museum database

and their originals (Figure 3.5 and 3.6)used in chapter 3 are all taken from the Victoria and Albert database, hence the CBIR experiment on QBLI will be evaluated only on that particular database. The *query by texture* on the other hand will be evaluated on all three databases, as it is quite clear from the previous section that the three databases are quite different in their image contents.

7.2.1 Query by Low-Quality Image

The 20 fax images used in chapter 3 are again used as query images. Recall that the QBLI gave a very good performance when used in a database size of 1062 images. In this experiment, the same procedures as in chapter 3 are used but on a much larger database size. The Victoria and Albert Museum database as described in the previous section consists of 16959 images. Six levels of decompositions are used to perform wavelet decomposition on 99 binaries per database image using Coiflet 6-tap wavelet basis, while the Manhattan distance metric are used to compute the dissimilarity measure. Table 7.1 shows the results of the experiment.

Clearly from the table, the proposed algorithm has no problems in adjusting to a larger database. All of the target originals are retrieved within the top 6. Considering there are almost 17000 different images within the database, top 6 performance is very good. In fact 13 of the 20 target originals are retrieved as the first ranked image, which suggests that the proposed algorithm manages to produce very similar feature vectors between the fax and the originals. Hence even with much more images added to the database to confuse the system, the algorithm will still be able to find the correct target.

Based on this observation, using any other kind of low-quality images described in chapter 3 should also be successful on large database. Finally the time taken to perform the retrieval is recorded to be around 80 seconds on average on a 700MHz Xeon processor. This is a reasonable performance considering there is no multidimensional indexing at all employed to index such a large number of feature vectors. An addition of a suitable

multidimensional indexing technique can help with the increasing size of database and improve the speed of retrieval, and make the CBIR system friendlier for the users. In addition, a suitable fast searching algorithm can also further improve the speed of the system.

7.2.2 Query by Texture

The multiscale-based and the segmentation-based approaches are both evaluated on the three museum databases using the procedures and parameters described in chapter 5 and 6 respectively. The three databases carry quite a different challenge. The Victoria and Albert Museum database has largely an image of an object, hence is not too complex, but the database size is the biggest. The National Gallery database contains more complex images because all of its images are paintings of a scene. Finally the C2RMF database offers the most difficult challenge where most of its images are quite a smudgy painting (because they are originally for use in restoration research) and also not many textures are significantly visible. Due to limitation in space, it is not possible to give too many examples of the retrieval performance of the algorithms. Nonetheless all the observations from the experiments are summarized below as well as suggested improvements to the algorithms. Figure 7.4, 7.5 and 7.6 show examples of retrieval results using both algorithms on the National Gallery, Victoria and Albert Museum and C2RMF databases respectively.

The most important observation from the experiments is that both algorithms work well with all three museum databases. Even for the C2RMF database, both algorithms manage to retrieve visually similar texture although its performance is not as good as for the other two databases. This is because the smudgy nature of images within the database brings a high level of confusion. The other two databases on the other hand show a much better retrieval result. Even for a database size of 16000 images for the Victoria and Albert Museum database, both *query by texture* algorithms manage to retrieve similar textures to the query. Both algorithms can be said to be applicable in the content-based image retrieval system.

The next observation is to compare the performance between the multiscale-based approach and the segmentation-based approach. From the experiments, it was observed that the multiscale-based approach gives better performance than the segmentation-based approach. One example can be seen from Figure 7.1. In this particular example, the multiscale-based approach manages to retrieve two images which have exactly the same texture as the query (images ranked first and third) while the segmentation-based approach can only manage to retrieve one of them. This is probably caused by a default in the segmentation process or the lack of multiscale property within the segmentation-based approach. It was also observed that the patches retrieved by the multiscale-based approach are also much more similar to the query. The segmentation-based approach

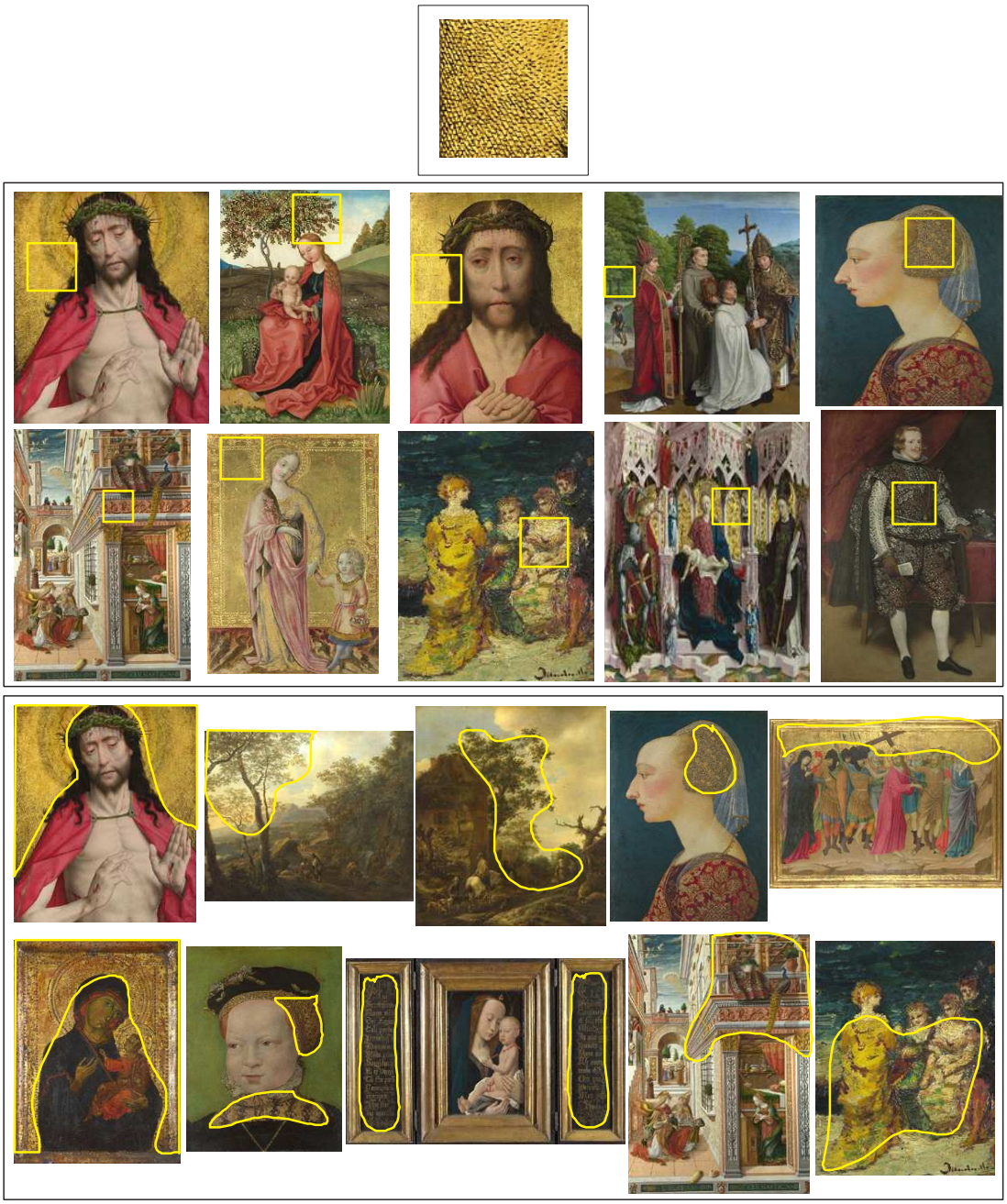


FIGURE 7.4: Retrieval example of National Gallery database (top) query image, (middle) result using multiscale-based approach, (bottom) result using segmentation-based approach

however has a much faster retrieval speed. This is because the number of feature vectors of the segmentation-based algorithm for a particular image is much lower than the multiscale-based algorithm. Overall, the advantages and disadvantages of both algorithms can be summarized as follows.

The multiscale-based approach has the advantage of much better accuracy at the expense of computational load. The multiscale nature of this approach also adds to its advantages as it has been proved to be useful in capturing both coarse and fine textures. The

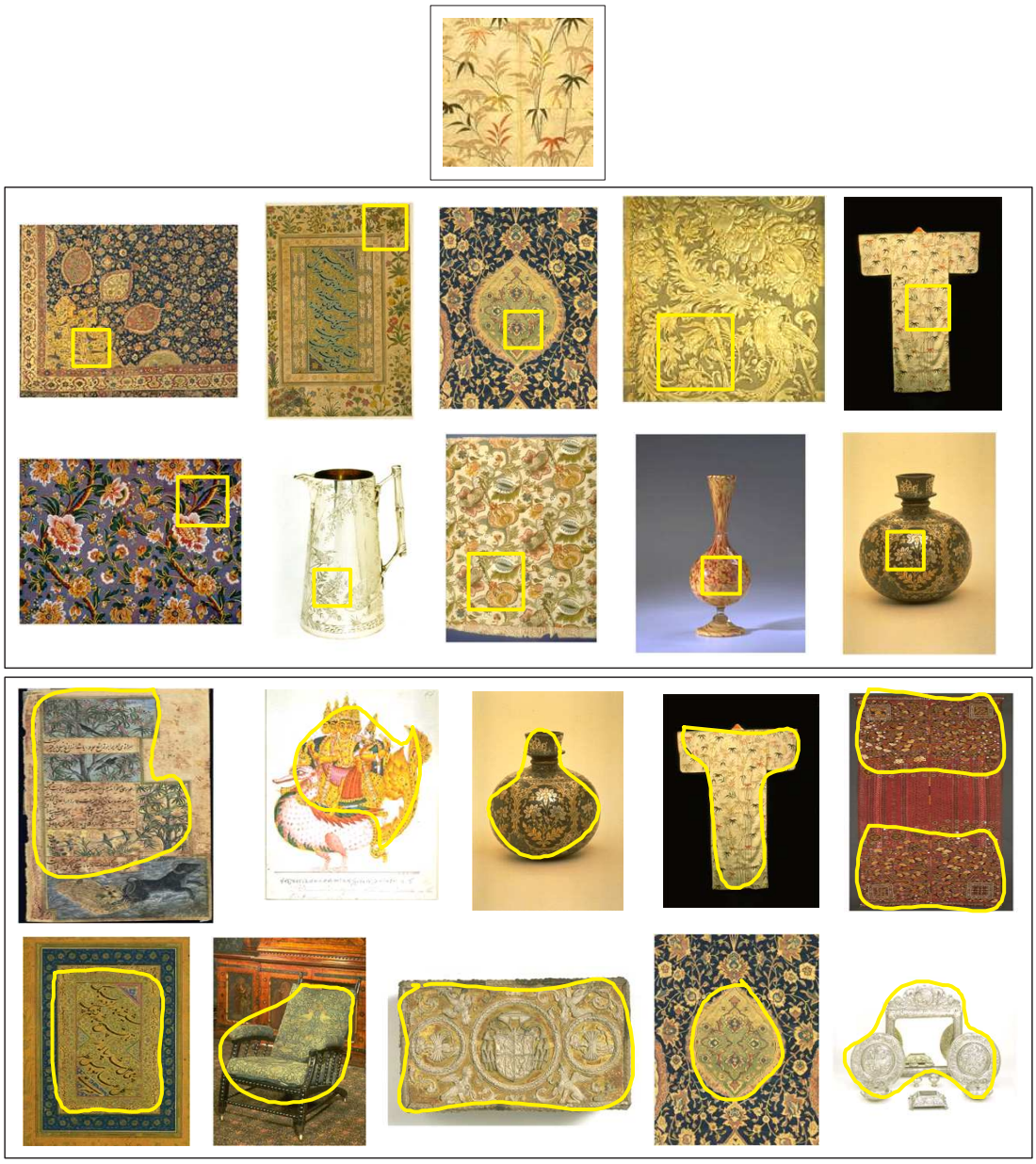


FIGURE 7.5: Retrieval example of Victoria and Albert Museum database (top) query image, (middle) result using multiscale-based approach, (bottom) result using segmentation-based approach

algorithm however can suffer from some odd retrieval result. For example when two textures are captured by a sub-image, the resulting feature vector might be similar to a feature vector of a completely different query texture, and hence will be retrieved as one of the top matches. However this is only a minor problem as it does not affect the overall performance of the algorithm. An improvement to the multiscale-based approach will mainly be on reducing the computational load. One of the possibilities is to use the *case 1 overlapping* for sub-image coverage instead of *case 2 overlapping* (see section 5.3.1), although the accuracy might drop quite drastically as well. Another possibility is to introduce the texture identifier like the one used for the segmentation process to

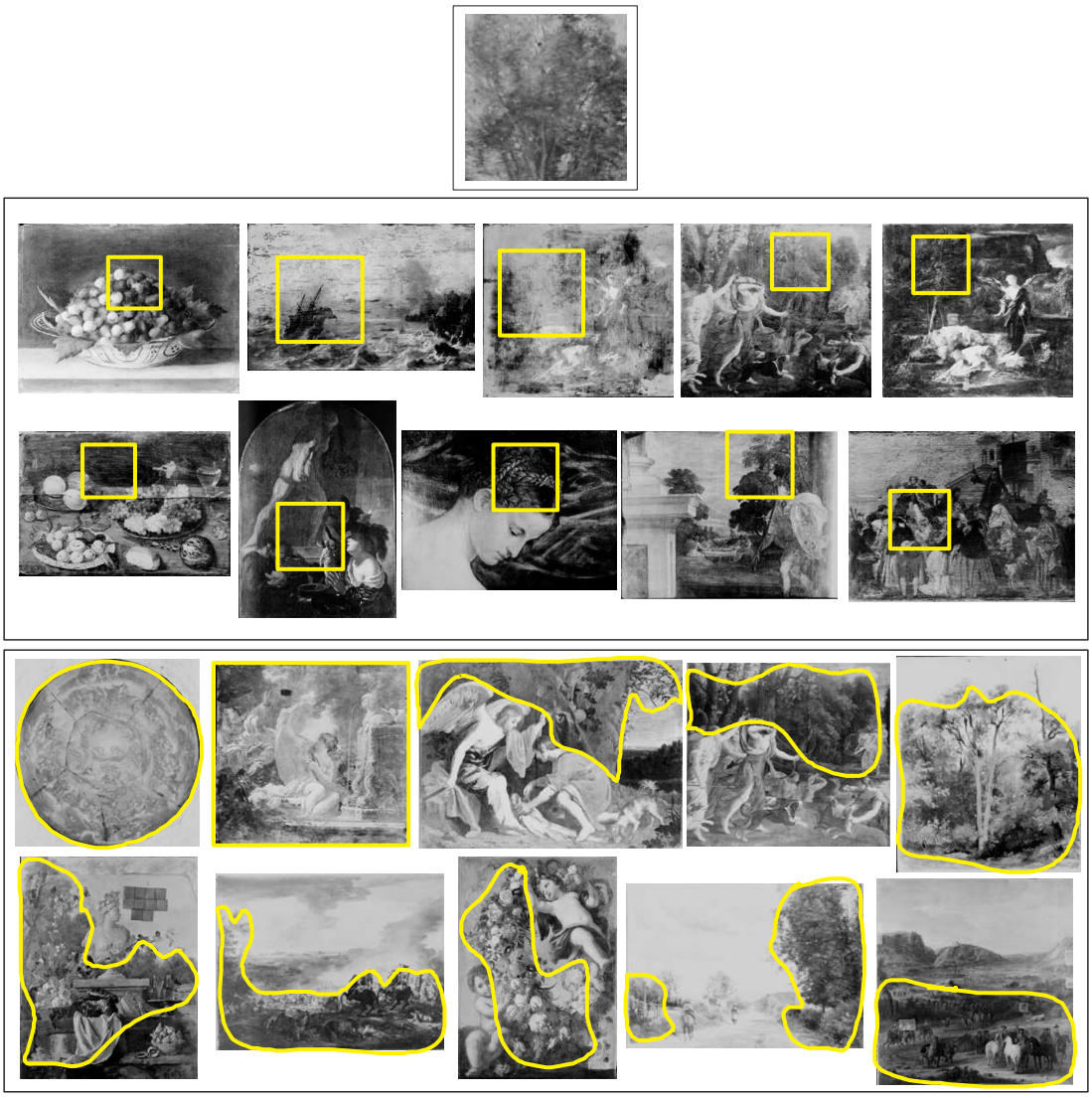


FIGURE 7.6: Retrieval example of C2RMF database (top) query image, (middle) result using multiscale-based approach, (bottom) result using segmentation-based approach

decide whether a particular sub-image is textured or not. The feature vectors are then created only for the textured patches, hence reducing the total number of feature vectors greatly. Finally a suitable multidimensional indexing algorithm can also be associated to help speed up the matching process.

The segmentation-based algorithm has the advantage of retrieval speed, although the accuracy is not as good as the multiscale-based approach. The computational load for the feature extraction process is also lower than that of the multiscale-based approach. Although this approach lacks the multiscale property, it can on the other hand provide the shape of the segmented objects. This might be useful for later purposes such as object identification or shape-from-texture processes. On the downside, this approach can suffer from a default in segmentation process, where either a texture is classified as insignificant or a texture is classified together with another texture. An improvement

to the algorithm means improving the texture segmenter itself to make it more accurate and reliable. If the multiscale property is desired, then a way for integrating it to the segmentation-based algorithm can be investigated.

Finally the choice between the multiscale-based and the segmentation-based approaches depends on application. If the user is willing to compensate a longer time for a better accuracy, then the multiscale-based approach will be suitable. However if the user wants a quick retrieval response and is willing to tolerate a slightly riskier outcome, then the segmentation-based approach will be more suitable. An algorithm which combines the advantages of the multiscale-approach with the advantages of the segmentation-based approach therefore needs to be explored.

7.3 Integration into the Artiste and Sculpteur Projects

Artiste (An Integrated Art Analysis and Navigation Environments) (146) is a European Commission supported project that has developed integrated content and metadata-based image retrieval across several major art galleries in Europe. Collaborating galleries include the four museums/galleries mentioned in the previous section. The Artiste system currently holds over 160,000 images from the separate collections owned by the mentioned partners. Artiste projects ran from the start of 2000 until the end of 2002. Over the three year period, Artiste has developed an image search and retrieval system that integrates distributed, heterogeneous image collections, providing a single interface to the art and its metadata.

The Artiste project addresses several problems which among others include:

- P1: Matching of similar images,
- P2: Search based on concept of style,
- P3: Search based on features oriented to restoration framework,
- P4: Access information quickly and easily,
- P5: Search based on colour,
- P6: Query by low quality images,
- P7: Query by sketch,
- P8: Joint retrieval by content and text,
- P9: Detail finding.

At the conclusion of the project in 2002, two out of the three proposed algorithms in this thesis has been successfully incorporated into the system. The *query by low-quality image* (P6) provides one of the main attributes of the system where, as mentioned before, the problem is quite important for the museums to address. The multiscale-based *query by texture* (P1 and P9) is the other algorithm successfully implemented within the system, albeit an earlier prototype algorithm, which uses the pyramidal wavelet transform instead of the discrete wavelet frames decomposition. The sub-image coverage is also not as detailed as the ones proposed in chapter 5. Nonetheless the successful integration of the *query by low-quality image* and the multiscale-based *query by texture* proves that both algorithms are suitable for an open interactive image retrieval of museum collections application.

The Artiste system also supports grid-based matching, which will be particularly useful in retrieving textures. As been mentioned before, a query image should not necessarily be a whole texture image. It could be an image of regular paintings or scenery, and only a certain part of the image contains the texture that we want to perform similarity retrieval. Using grid-based matching, the image is divided into equally sized regions, and the interface allows selection and deselection of elements of the grid within the image. A selected element implies that it is a pertinent area containing features which are to be searched for. Figure 7.7 illustrates this process.

FIGURE 7.7: Grid-based matching, where the regions to be searched for is selected

Compared to the other CBIR systems such as the QBIC, Virage, Photobook etc as mentioned in section 2.1.2, the major difference between those systems and the value of Artiste is that Artiste is specifically produced for the purposes of the museums and galleries, and therefore has some functionalities which other systems do not provide. Artiste has some things in common with these other systems. For example, all the systems, including Artiste, have the ability to search for images based on their colour distribution, and many of them also have some whole image texture measure. Artiste on the downside lacks the shape-based querying of QBIC, Virage, RetrievalWare and

Photobook, but is the only one of the systems that allows the matching of queries for sub-images (with the multiscale-based approach) and is the only system to facilitate dynamic links and navigations.

Sculpteur (Semantic and Content-based Multimedia Exploitation for European Benefit) (147) is a continuation project to Artiste. It is a three-year European project that started in May 2002 and is co-funded by the European Community. In addition to the four museums collaborating in the Artiste project is the Museum of Cherbourg in France. The Sculpteur project will further develop and improve the Artiste system, including the concept and content-based retrieval application, while developing the technology and expertise to help create, manage and present cultural archives of 3D models and associated multimedia objects.

The texture algorithms from Artiste are being imported to Sculpteur and it is likely that with some further development they will also be used for 3D object retrieval via the texture maps of these objects. Finally, as part of the work, Sculpteur also exploits an exciting new development in web technology, the semantic web, and develops e-Learning systems able to use the semantic knowledge created during the project.

Chapter 8

Conclusion and Future Work

This research has been concerned with the development of algorithms to tackle some of the problems faced in the content-based image retrieval of museum image collections, namely *query by low-quality image* and *query by texture*. A new methodology for retrieving images based on low-quality queries is proposed and developed, while two new methodologies are also proposed, developed and compared to solve the problem of texture retrieval for museum collections.

Image retrieval based on low-quality query is a very challenging task and poses some crucial problem for the museum curator in the form of fax images. There are not many published works on this type of problem available in the literature which makes it an even more challenging field. The research began by tackling the underlying properties of the fax images and how several modifications on the available *query by image example* could improve the retrieval performance. Query by texture on the other hand is quite a well documented research area. The problem however is how to capture the property of local texture instead of the global ones. Two approaches namely the multiscale-based approach and the segmentation-based approach are proposed and evaluated for use in the vast museum collections.

This chapter presents the main innovations of this research by summarizing the three proposed techniques presented in chapter 3, 4, 5 and 6, as well as the performance observation on an interactive content-based image retrieval application in chapter 7.

8.1 Conclusion

The overall aim of this research was to address the problems of *query by low-quality image* and *query by texture*, which was a requirement for the Artiste project. The work on the *query by low-quality image* began with identifying the binary nature of the main source of low-quality images for the museums, the fax images. It was found that the almost

black and white fax images lost most of their information at the receiving end of the fax machine; hence a standard query by image example search will be inadequate to retrieve correct target images. A novel *query by low-quality image* technique which uses a binary matching algorithm together with the pyramidal wavelet transform is proposed. For each image, the algorithm creates a certain number of binaries by adjusting the threshold for binarisation, and the pyramidal wavelet transform is used to extract features from the binaries. During matching, the percentage of black pixels within the binary version of the query is computed and its feature vector is compared with only one of the many feature vectors associated with each database image, namely the feature vector corresponding to the binary image that has the closest percentage of black pixels to the query binary.

A simple but highly time consuming technique which is called the pixel matching technique was also proposed and developed, to be used as a yardstick for the evaluation of the proposed QBLI algorithm. The pixel matching technique gave very good retrieval accuracy, while the proposed QBLI algorithm gave a comparable performance. This implies that the QBLI algorithm is almost as good as the pixel matching algorithm, but has the advantage of being much faster computationally, making it more suitable for interactive use. It was also found that the proposed binarisation stage of the algorithm and the pyramidal wavelet transform is very important as they are the main factors in improving the accuracy and keeping a minimal computational load respectively. Finally the proposed algorithm lends itself quite well to solving query by other low-quality image types. Experiments showed that good results are obtained in solving images of inappropriate brightness and contrast, highly compressed images, low-resolution images, quantized images and noisy images.

The *query by texture* research began by identifying the best texture feature extraction techniques to be used in extracting local texture features. It was found that the best overall technique is the discrete wavelet frames decomposition. The best parameters to be associated with the discrete wavelet frames were also identified in order to further improve the retrieval accuracy of the technique. Now that the local texture feature extraction is identified, two approaches for *query by texture*, the multiscale-based and the segmentation-based approach were proposed and evaluated. The two approaches were considered because it is also interesting to observe whether actual texture segmentation is really needed in order to perform *query by texture*, or whether the block-based approach is just as good.

The multiscale-based approach offers a better localisation compared to other block-based technique because it has the multiscale property which can reduce the scale-dependence of the discrete wavelet frames texture features. Some modifications are proposed for the multiscale algorithm in order to make it suitable for texture retrieval. The modifications include the scaling factor for different scales, as well as the positioning of the blocks within the entire image of interest. The multiscale-based approach was evaluated on several databases with known ground truth in order to observe the coverage of its sub-

images and the effect of different query sizes and scales to the algorithm. It was observed that the approach is very suitable for texture retrieval of museum collections.

For the segmentation-based approach, a new framework for automatic texture segmentation based on modified discrete wavelet frames and the mean shift algorithm is developed. The texture segmenter consists of a top-down decomposition phase and a bottom-up segmentation phase. Top-down decomposition works by first performing a discrete wavelet frames decomposition on the image to be segmented, followed by some feature averaging to reduce the amount of data to be clustered. In the bottom-up segmentation, the mean shift algorithm is used to detect the number of segments at each level, and the fuzzy c-means clustering is used to assign each pixel to the desired segments. The process continues until the root level is reached and the final segmentation is obtained.

By modifying the discrete wavelet frames, much better clustering is obtained on a reduced set of data, making possible the use of the mean shift algorithm to detect the correct number of clusters, and substantially reduces the processing time. The mean shift also provides the position of the cluster centres which effectively solves the problem of initializing the membership function in the fuzzy c-means algorithm, and hence reducing the fuzzy iterations. It was observed that the proposed segmentation method can detect the correct number of clusters as well as segmenting the image correctly in composite textures, synthetic textures, real scene images and museum images. The texture segmentation was then extended to be a texture identifier in order to use it in an image retrieval system, where only textured regions are processed for feature extraction. It was observed that the segmentation-based retrieval system performs well in retrieving similar texture from a museum database, hence it provides an alternative to the multiscale-based texture retrieval.

All three algorithms were then evaluated on several different museum databases with much larger sizes. For the *query by low-quality image*, the algorithm works well even for a database size of 16,000 images, which further proves the reliability of the proposed technique. The same was observed for the *query by texture* algorithms, where promising performance was recorded. It was also observed that the multiscale-based approach is better than the segmentation-based algorithm in terms of retrieval accuracy although it comes with a much slower retrieval speed. The choice of which algorithm is more suitable thus depends on the preference of the user. Finally the *query by low-quality image* and the multiscale-based *query by texture* have been successfully integrated with the Artiste and Sculpteur image retrieval system.

8.2 Future Work

The three algorithms presented in this thesis successfully demonstrated innovation and novelty. The results that have been obtained are highly promising and have reasonably

demonstrated a proof of concept. However there is substantial scope for further research and development to the system.

For the *query by low-quality image*, as mentioned in chapter 3, a problem might occur if the object to background ratio of the image differs significantly from the target images. In this case, a special pre-processing algorithm to detect plain background can be developed so that all images will have their default object to background ratio, hence providing the solution. A suitable indexing method can also be introduced in order to improve the retrieval speed when the database size increases by several factors.

The improvement on the multiscale-based *query by texture* should mainly be on introducing a suitable multidimensional indexing technique to assist the high volume of feature vectors the algorithm produces. The Artiste image retrieval system, which also does not incorporate any multidimensional indexing technique, will most probably be very slow when using the final prototype of the multiscale-based approach because of the high volume of images in the database as well as the very high resolution for each image. An introduction of a texture verifier for each sub-image and a suitable indexing algorithm will help improve the speed.

Future works on the segmentation-based *query by texture* can be in improving the segmentation accuracy as this will reduce the possibility of pixels from another segment affecting the outcome of the feature vector, as well as reducing the possibility of wrongly segmented regions. Moreover it can also provide a better shape for use in extracting shape features for further process, if needed. If the multiscale property is desired, then a way for integrating the property to the segmentation-based algorithm can be investigated.

Finally all of the proposed algorithms in this thesis could be extended to provide semantic content-based image retrieval. Working towards a semantics-based content-based image retrieval engine is a very enticing goal. If the semantic gap was bridged effectively enough to allow such a retrieval system, the way we search for images could change dramatically. This is what the Sculpteur project is trying to achieve, and is very worthwhile to look at.

Appendix A

Comparison of Texture Features Performance

Pyramidal Wavelet Transform

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	95	100	100
D002	44.17	54.17	58.75
D003	44.17	59.58	72.92
D004	68.75	84.58	92.5
D005	67.5	83.33	91.67
D006	99.58	100	100
D007	35	45	51.67
D008	94.17	100	100
D009	48.75	70.42	77.92
D010	61.25	70	74.58
D011	90	97.08	98.75
D012	75	78.33	80.83
D013	29.58	46.67	55.42
D014	100	100	100
D015	56.67	61.67	65.42
D016	100	100	100
D017	100	100	100
D018	86.25	96.25	97.5
D019	82.5	90.42	96.67
D020	100	100	100
D021	99.58	99.58	100
D022	47.5	57.08	61.67
D023	40.42	67.08	77.08
D024	98.33	100	100
D025	49.58	65	72.5
D026	88.33	100	100
D027	46.67	72.92	85.42
D028	56.67	71.25	80.83
D029	93.75	100	100
D030	32.5	59.58	73.75
D031	53.33	67.5	72.5
D032	94.17	98.33	99.58
D033	83.33	90.42	95.42
D034	98.33	100	100
D035	64.17	80.83	90
D036	64.17	86.25	89.17
D037	70	80.83	87.08
D038	50.83	55.83	57.92
D039	42.92	49.58	59.58
D040	34.17	44.58	50.83
D041	76.67	92.08	95.83
D042	26.25	28.75	34.17
D043	9.17	11.25	14.17
D044	6.67	8.75	9.58
D045	7.08	9.17	10.42
D046	81.67	91.67	95
D047	100	100	100
D048	99.58	100	100
D049	100	100	100
D050	72.5	92.92	96.25
D051	96.67	100	100
D052	54.58	56.25	57.92
D053	100	100	100
D054	35	41.25	45.83
D055	100	100	100
D056	96.67	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	96.67	100	100
D058	15.83	25	31.67
D059	20	29.17	36.25
D060	34.17	46.67	54.58
D061	47.08	58.75	67.08
D062	65.83	81.25	86.25
D063	42.5	55.42	62.92
D064	94.17	99.58	100
D065	99.58	100	100
D066	71.25	85.42	91.67
D067	47.08	56.67	65
D068	100	100	100
D069	34.58	41.67	47.08
D070	59.58	74.17	88.33
D071	68.33	89.58	97.08
D072	72.92	80.83	85.83
D073	35	49.17	58.33
D074	74.58	89.58	95
D075	88.75	97.92	99.17
D076	99.17	99.58	100
D077	100	100	100
D078	87.08	95	96.67
D079	100	100	100
D080	77.92	99.17	100
D081	87.08	98.75	99.58
D082	95.42	100	100
D083	90	99.58	100
D084	98.75	99.58	100
D085	60.83	72.92	78.75
D086	55.42	74.17	85
D087	94.17	100	100
D088	43.75	69.17	81.25
D089	27.92	42.92	55
D090	27.5	35.83	41.25
D091	39.17	55.42	63.75
D092	75.42	99.58	100
D093	54.58	68.75	76.67
D094	91.25	100	100
D095	59.17	69.17	77.08
D096	55.42	68.75	76.25
D097	22.08	34.58	46.25
D098	77.08	97.5	99.58
D099	42.92	70.42	90
D100	36.25	50.83	60.42
D101	67.92	90.42	95.83
D102	53.33	83.75	93.75
D103	70	98.75	100
D104	81.67	100	100
D105	47.08	93.75	100
D106	47.08	98.33	100
D107	49.17	67.5	78.75
D108	13.75	20.42	24.58
D109	35	52.5	63.75
D110	38.75	66.25	82.08
D111	66.67	85.83	90.42
D112	46.67	56.25	62.5

Tree-Structured Wavelet Transform

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	90	98.75	100
D002	50	59.17	61.25
D003	65.83	78.33	85.83
D004	73.33	90.83	95.42
D005	48.75	58.33	66.67
D006	100	100	100
D007	32.92	39.58	44.58
D008	92.08	99.17	99.58
D009	42.08	65.42	83.75
D010	66.25	75	81.67
D011	91.25	96.67	97.5
D012	69.58	73.75	78.75
D013	25.42	37.5	43.33
D014	100	100	100
D015	54.58	60.83	65
D016	100	100	100
D017	100	100	100
D018	93.75	98.33	99.58
D019	78.75	90	95.42
D020	100	100	100
D021	100	100	100
D022	54.58	60.83	63.75
D023	41.25	60.42	72.92
D024	92.08	98.75	100
D025	63.75	75.83	82.92
D026	92.92	99.17	100
D027	51.25	77.5	88.33
D028	56.67	72.92	81.67
D029	93.33	100	100
D030	42.92	62.92	75.42
D031	42.08	56.25	61.67
D032	82.92	92.5	97.92
D033	86.67	93.33	96.67
D034	93.33	100	100
D035	57.08	76.25	84.17
D036	47.5	59.58	67.08
D037	74.17	84.58	89.17
D038	41.67	47.5	52.5
D039	42.08	49.17	55.42
D040	40	47.5	52.5
D041	76.25	89.58	93.33
D042	31.25	35.83	39.58
D043	8.33	10.83	12.08
D044	9.17	10.83	13.75
D045	4.17	5.42	7.5
D046	62.92	76.67	84.58
D047	100	100	100
D048	100	100	100
D049	100	100	100
D050	87.5	94.58	97.92
D051	92.92	98.75	100
D052	56.67	57.92	58.75
D053	100	100	100
D054	30	30.42	30.83
D055	100	100	100
D056	100	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	98.33	100	100
D058	21.67	27.5	35
D059	25.42	35.42	42.92
D060	41.67	60	69.17
D061	48.75	62.5	69.58
D062	66.67	74.58	78.75
D063	49.58	64.17	70.42
D064	93.75	98.33	100
D065	100	100	100
D066	60.42	77.08	87.5
D067	52.08	65.83	70.83
D068	100	100	100
D069	48.75	64.17	66.67
D070	68.75	86.25	97.5
D071	80.42	96.25	98.33
D072	54.58	68.33	77.08
D073	27.92	37.5	45.83
D074	67.5	80	87.08
D075	84.17	96.25	99.17
D076	96.25	100	100
D077	100	100	100
D078	95.83	99.17	99.58
D079	100	100	100
D080	88.33	100	100
D081	90.42	98.75	99.17
D082	100	100	100
D083	99.58	100	100
D084	97.08	99.58	100
D085	86.25	91.67	94.58
D086	64.17	80.83	86.67
D087	87.92	97.92	99.17
D088	39.58	60.83	70.42
D089	27.92	43.33	55.42
D090	31.25	38.33	42.92
D091	51.25	67.92	77.5
D092	75.42	93.33	97.08
D093	43.75	53.33	56.67
D094	92.08	99.17	100
D095	81.25	93.75	97.08
D096	64.17	73.33	77.08
D097	29.58	47.08	58.33
D098	67.92	90.42	98.33
D099	40.42	69.17	90
D100	23.33	31.67	41.25
D101	68.75	89.17	93.75
D102	57.08	77.92	91.25
D103	66.67	98.33	100
D104	69.58	99.17	100
D105	55.42	99.17	100
D106	40.42	91.67	99.58
D107	50.42	64.58	72.92
D108	13.33	16.67	22.5
D109	32.5	52.08	66.25
D110	30	62.92	84.58
D111	50.83	68.33	75.42
D112	48.33	53.75	58.75

Discrete Wavelet Frames

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	94.58	100	100
D002	46.25	54.58	59.17
D003	47.08	63.33	75.42
D004	69.17	85	90.83
D005	67.92	83.33	91.67
D006	100	100	100
D007	35.83	46.67	52.92
D008	94.17	100	100
D009	48.75	69.17	77.92
D010	62.5	71.25	76.25
D011	93.75	99.17	99.17
D012	75.83	79.17	81.25
D013	29.58	44.17	52.92
D014	100	100	100
D015	56.67	62.92	65
D016	100	100	100
D017	100	100	100
D018	86.67	98.75	100
D019	80	87.92	95.83
D020	100	100	100
D021	100	100	100
D022	48.33	58.33	61.67
D023	44.17	67.5	81.67
D024	98.75	100	100
D025	47.5	64.58	73.33
D026	98.33	100	100
D027	49.17	73.75	84.17
D028	62.5	74.17	80.83
D029	93.33	100	100
D030	35.42	60.42	73.33
D031	52.5	67.92	73.75
D032	94.58	99.17	100
D033	82.92	90.42	96.67
D034	97.92	100	100
D035	68.75	84.17	90.42
D036	63.33	83.33	89.17
D037	73.75	83.33	89.17
D038	52.08	57.08	58.33
D039	42.92	50	60
D040	32.92	45	51.25
D041	78.33	92.92	97.5
D042	24.58	30	33.33
D043	9.17	11.25	14.58
D044	6.67	8.33	10
D045	7.08	9.58	10.83
D046	79.58	90.42	94.17
D047	100	100	100
D048	99.17	100	100
D049	100	100	100
D050	74.17	93.75	96.67
D051	97.5	100	100
D052	52.92	56.25	57.5
D053	100	100	100
D054	35.42	40.83	43.33
D055	100	100	100
D056	98.33	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	98.75	100	100
D058	17.92	27.08	32.5
D059	20.42	29.58	38.33
D060	35.42	45	53.75
D061	46.67	59.17	68.75
D062	66.25	80.42	86.67
D063	42.5	52.92	62.08
D064	95.42	99.58	100
D065	100	100	100
D066	73.33	85.42	90.42
D067	47.08	56.67	64.17
D068	100	100	100
D069	33.33	41.25	45.42
D070	59.17	73.75	89.58
D071	69.17	89.58	97.08
D072	74.17	80.42	85
D073	36.25	49.17	59.17
D074	73.75	87.92	93.33
D075	89.17	98.75	99.17
D076	98.75	100	100
D077	100	100	100
D078	89.58	95	96.67
D079	100	100	100
D080	76.67	99.58	100
D081	86.67	98.33	99.58
D082	95.42	100	100
D083	93.75	100	100
D084	98.75	100	100
D085	61.25	70.83	77.08
D086	58.33	74.58	86.25
D087	92.92	100	100
D088	42.5	69.58	82.5
D089	27.92	43.33	53.75
D090	27.08	35.42	40
D091	41.25	56.25	62.92
D092	78.75	99.58	100
D093	57.08	71.67	81.25
D094	97.92	100	100
D095	60.83	70.42	76.25
D096	56.67	68.33	75
D097	21.67	36.25	46.67
D098	74.58	97.92	99.58
D099	41.25	75.83	91.67
D100	39.17	51.67	62.08
D101	76.25	95.42	100
D102	80	99.17	100
D103	74.58	99.17	100
D104	81.25	100	100
D105	47.08	94.58	100
D106	45.83	98.33	100
D107	50	70.42	79.17
D108	12.08	21.67	25.83
D109	32.92	51.67	63.75
D110	40.83	67.08	81.25
D111	67.5	86.67	90.83
D112	44.58	55	62.92

Gabor Transform with 6 Orientations and 3 Scales

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	94.58	99.17	99.58
D002	52.92	58.33	60.83
D003	91.67	98.75	99.58
D004	99.58	100	100
D005	74.58	88.33	92.92
D006	100	100	100
D007	40	50	57.5
D008	94.17	100	100
D009	66.67	85.42	93.75
D010	69.17	77.5	83.33
D011	100	100	100
D012	78.33	83.33	86.25
D013	51.67	62.08	69.58
D014	100	100	100
D015	56.67	62.92	67.92
D016	100	100	100
D017	100	100	100
D018	95	99.17	100
D019	80	90.83	95
D020	100	100	100
D021	100	100	100
D022	46.25	58.75	66.67
D023	30.83	48.33	61.25
D024	93.75	98.33	98.33
D025	49.17	62.5	70.83
D026	98.33	99.58	100
D027	39.58	56.25	65
D028	63.33	75.83	80.42
D029	96.25	100	100
D030	27.08	39.58	46.25
D031	42.92	60.42	70
D032	99.58	100	100
D033	87.08	96.25	97.5
D034	99.58	100	100
D035	60.42	70	75.42
D036	54.58	75.42	84.58
D037	91.25	99.17	100
D038	54.58	64.17	72.5
D039	41.25	52.08	59.58
D040	27.92	39.58	47.92
D041	65.83	83.33	91.67
D042	21.67	28.75	32.5
D043	12.92	15.42	16.67
D044	7.08	10	10.83
D045	2.08	4.17	4.58
D046	97.08	100	100
D047	96.25	98.33	99.58
D048	64.17	78.75	83.33
D049	100	100	100
D050	68.75	84.58	87.92
D051	99.17	100	100
D052	63.33	66.25	72.08
D053	100	100	100
D054	39.17	51.25	58.75
D055	100	100	100
D056	99.58	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	100	100	100
D058	14.58	26.25	32.5
D059	10.83	16.67	22.08
D060	33.75	47.08	53.75
D061	28.33	43.33	53.33
D062	32.5	45.42	56.67
D063	27.08	37.08	45
D064	95	100	100
D065	100	100	100
D066	57.08	70.42	79.58
D067	54.17	60.42	64.17
D068	100	100	100
D069	29.58	36.67	40.83
D070	53.75	62.92	69.17
D071	47.92	69.58	83.33
D072	63.33	77.5	85.83
D073	43.75	60.83	70.42
D074	84.17	97.5	99.17
D075	84.58	93.75	95.42
D076	96.25	100	100
D077	100	100	100
D078	99.17	100	100
D079	100	100	100
D080	99.58	100	100
D081	97.92	100	100
D082	93.75	100	100
D083	98.33	100	100
D084	99.17	100	100
D085	72.5	86.67	89.58
D086	82.5	91.67	95.42
D087	89.17	100	100
D088	25.83	43.75	51.67
D089	17.5	29.17	35.42
D090	16.67	23.33	28.33
D091	14.17	19.58	25.83
D092	93.33	99.17	100
D093	77.92	85.83	88.75
D094	97.92	100	100
D095	58.33	65.42	72.08
D096	47.92	55.83	63.75
D097	18.33	25.42	32.5
D098	62.5	85.42	95.83
D099	27.5	49.17	62.5
D100	44.58	58.75	70
D101	86.67	99.17	100
D102	81.67	100	100
D103	63.75	98.33	100
D104	63.33	100	100
D105	55.83	96.67	100
D106	47.08	91.67	100
D107	47.08	66.25	76.67
D108	12.08	19.17	22.92
D109	32.08	52.5	67.92
D110	41.67	69.58	82.5
D111	55	69.58	77.92
D112	42.08	52.08	56.25

Gabor Transform with 6 Orientations and 4 Scales

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	97.92	100	100
D002	57.08	59.58	62.08
D003	93.75	98.75	99.58
D004	100	100	100
D005	81.25	95.83	98.75
D006	100	100	100
D007	41.67	52.08	59.17
D008	96.25	99.17	100
D009	62.08	80.42	91.67
D010	69.58	79.17	83.33
D011	100	100	100
D012	81.67	86.67	87.92
D013	49.58	63.33	69.58
D014	100	100	100
D015	57.5	64.58	66.67
D016	100	100	100
D017	100	100	100
D018	99.17	100	100
D019	82.92	93.33	97.5
D020	100	100	100
D021	100	100	100
D022	50.42	71.25	78.33
D023	55.83	77.08	83.33
D024	90	97.08	98.33
D025	68.33	80.42	90
D026	100	100	100
D027	52.5	72.5	81.25
D028	69.58	82.5	88.75
D029	97.5	100	100
D030	45.42	62.92	70.42
D031	31.67	49.17	62.08
D032	99.58	100	100
D033	86.25	93.33	97.08
D034	100	100	100
D035	59.58	66.25	73.75
D036	50.42	71.67	77.5
D037	89.58	96.25	98.33
D038	57.08	67.92	73.33
D039	47.92	58.75	66.25
D040	29.17	42.08	54.17
D041	65.83	84.17	89.58
D042	23.33	29.17	33.75
D043	9.58	13.75	17.08
D044	10.83	12.5	15.83
D045	5.42	5.83	7.92
D046	96.25	100	100
D047	98.33	100	100
D048	86.25	90	100
D049	100	100	100
D050	83.75	95.42	97.08
D051	91.67	99.17	100
D052	74.17	77.92	82.5
D053	100	100	100
D054	37.5	47.5	53.75
D055	100	100	100
D056	100	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	100	100	100
D058	20	34.17	43.33
D059	20.83	32.92	40.42
D060	40.42	55.42	65
D061	36.25	50	67.08
D062	48.33	65.42	76.67
D063	37.92	49.17	59.58
D064	94.17	100	100
D065	100	100	100
D066	54.17	73.33	80
D067	53.75	63.33	69.17
D068	100	100	100
D069	34.17	42.92	47.92
D070	63.33	83.33	90.83
D071	55	82.92	93.75
D072	54.17	68.33	74.58
D073	42.08	60	68.75
D074	86.25	97.92	99.58
D075	95.42	100	100
D076	98.33	100	100
D077	100	100	100
D078	100	100	100
D079	100	100	100
D080	100	100	100
D081	98.75	100	100
D082	100	100	100
D083	100	100	100
D084	100	100	100
D085	89.58	95.42	97.92
D086	88.33	97.5	99.58
D087	90	100	100
D088	26.25	45.83	62.08
D089	25.42	40.83	52.5
D090	26.67	32.08	39.17
D091	43.75	53.75	60.83
D092	89.17	98.75	100
D093	81.67	90	93.33
D094	98.75	100	100
D095	68.75	78.75	85.83
D096	61.25	72.08	78.75
D097	19.58	30	35.42
D098	60.83	87.92	98.75
D099	31.25	60.42	80.83
D100	41.25	56.67	68.75
D101	82.08	99.58	100
D102	73.33	98.75	100
D103	62.5	97.08	100
D104	60.42	99.58	100
D105	57.5	98.75	100
D106	45.83	91.25	100
D107	43.75	60.42	72.92
D108	17.5	22.5	27.5
D109	32.92	54.17	70
D110	43.33	72.5	84.58
D111	60.42	76.25	85.83
D112	44.58	58.75	63.75

Discrete Cosine Transform

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	95.83	99.58	100
D002	32.08	42.08	47.08
D003	38.75	55.42	67.08
D004	45.42	69.58	87.08
D005	31.67	42.08	49.17
D006	90	99.17	99.58
D007	16.67	21.25	24.58
D008	80.42	91.67	93.75
D009	41.67	70	92.92
D010	37.92	51.25	61.25
D011	67.92	82.5	89.58
D012	54.17	57.08	62.5
D013	9.58	17.5	24.17
D014	100	100	100
D015	46.25	55.83	60
D016	100	100	100
D017	91.25	99.17	99.58
D018	59.17	77.92	85
D019	51.25	62.92	72.92
D020	100	100	100
D021	100	100	100
D022	25.42	31.67	34.17
D023	27.92	45.42	55
D024	89.17	96.67	98.33
D025	54.58	60.42	65.83
D026	79.17	87.5	92.08
D027	43.33	64.17	76.67
D028	41.25	58.33	71.25
D029	70	88.75	96.25
D030	31.67	50	62.5
D031	19.58	31.25	41.67
D032	46.25	67.92	78.33
D033	51.25	66.25	74.58
D034	77.08	87.92	91.25
D035	39.17	55.83	68.33
D036	35	49.17	60.83
D037	61.25	71.25	78.33
D038	30	36.25	43.75
D039	33.33	40.83	43.33
D040	26.67	30.42	34.58
D041	57.92	71.67	80.83
D042	17.5	24.58	31.67
D043	5.83	6.25	8.33
D044	7.5	9.17	10.42
D045	4.17	6.25	7.08
D046	53.33	71.67	79.17
D047	70	82.92	90.42
D048	85	100	100
D049	100	100	100
D050	55.42	71.67	80
D051	72.5	83.75	87.92
D052	33.33	35.42	40.83
D053	99.58	100	100
D054	24.58	27.92	29.17
D055	92.5	98.75	100
D056	98.33	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	79.58	92.08	96.25
D058	23.33	29.17	32.08
D059	20.42	34.17	42.08
D060	29.58	42.92	51.25
D061	31.25	43.33	49.17
D062	45.83	56.67	65
D063	25	38.33	44.58
D064	96.67	99.17	99.17
D065	98.75	100	100
D066	44.58	60	71.67
D067	29.17	36.67	45
D068	96.25	100	100
D069	36.25	50	55.83
D070	61.25	75.42	86.25
D071	72.08	86.25	91.67
D072	43.33	59.58	66.25
D073	19.58	30.42	33.75
D074	34.58	45.83	55.83
D075	49.17	69.17	79.58
D076	75	90.83	96.25
D077	51.25	72.5	81.25
D078	67.5	75	81.25
D079	92.5	97.92	99.58
D080	70.42	90.83	94.58
D081	72.08	85	91.67
D082	88.33	100	100
D083	65.42	75.83	79.17
D084	79.58	89.17	95
D085	55.83	64.17	67.5
D086	50.42	66.25	74.58
D087	70	86.67	92.5
D088	23.33	39.58	54.58
D089	17.92	30.83	43.75
D090	22.5	27.5	32.5
D091	55	67.92	77.92
D092	50	71.67	82.08
D093	39.58	45.83	49.58
D094	81.25	93.33	96.67
D095	64.17	71.67	81.25
D096	44.58	55	62.5
D097	17.08	30.42	36.25
D098	47.08	70.83	86.25
D099	37.92	58.33	74.58
D100	14.58	18.75	25
D101	49.58	65.42	77.08
D102	70.42	90.83	97.08
D103	69.17	98.75	100
D104	75.42	99.17	100
D105	53.75	96.67	99.17
D106	41.25	94.58	100
D107	14.58	30	42.08
D108	8.33	13.75	17.08
D109	23.33	40.42	51.67
D110	36.67	57.92	70.83
D111	39.58	50	57.5
D112	25	32.08	39.17

Law's Texture Feature

Texture ID	Top matches considered			Texture ID	Top matches considered		
	Top 15	Top 30	Top 45		Top 15	Top 30	Top 45
D001	97.08	100	100	D057	78.33	95	97.92
D002	43.33	52.5	58.33	D058	23.33	27.5	33.33
D003	74.17	87.92	91.67	D059	30.42	40	47.5
D004	74.58	87.92	93.33	D060	35.83	50.83	59.17
D005	36.25	44.17	50.42	D061	46.67	56.25	63.75
D006	100	100	100	D062	46.67	62.5	68.75
D007	18.75	25.42	27.5	D063	39.58	52.08	60.83
D008	87.92	93.75	98.33	D064	97.08	100	100
D009	51.25	80.42	97.08	D065	100	100	100
D010	52.08	66.25	72.5	D066	65.42	87.08	93.33
D011	77.92	90.42	95	D067	35	42.08	46.25
D012	55	57.92	65.83	D068	98.33	100	100
D013	14.58	22.08	31.25	D069	48.75	58.75	64.17
D014	100	100	100	D070	62.92	80.42	87.92
D015	54.58	61.25	65.42	D071	84.58	93.75	96.67
D016	100	100	100	D072	47.08	58.33	65.83
D017	96.67	100	100	D073	22.92	32.08	39.58
D018	76.67	88.33	92.5	D074	40.42	52.08	63.75
D019	64.17	79.58	84.17	D075	55	76.25	85.83
D020	100	100	100	D076	87.08	97.08	98.75
D021	100	100	100	D077	80.42	89.17	93.33
D022	33.75	39.17	42.5	D078	87.08	92.08	92.92
D023	37.92	52.08	63.75	D079	95.83	99.58	100
D024	92.5	97.5	99.17	D080	82.08	94.17	95.83
D025	70	83.33	88.75	D081	74.17	85.83	90.42
D026	88.75	92.5	94.58	D082	96.25	100	100
D027	49.17	75.83	85.83	D083	76.67	82.5	87.92
D028	57.5	71.67	82.08	D084	95.83	97.92	99.17
D029	80.42	97.08	100	D085	65.83	76.25	77.92
D030	31.25	48.33	62.92	D086	62.92	74.17	82.5
D031	22.92	37.08	45	D087	72.5	89.17	94.58
D032	58.33	84.17	93.33	D088	32.5	55.42	68.33
D033	67.92	85.83	92.92	D089	32.5	44.17	54.17
D034	91.67	97.08	98.33	D090	20.83	30	34.17
D035	42.92	59.17	70.42	D091	54.58	69.58	79.58
D036	45	62.92	70.83	D092	66.67	84.58	90
D037	69.58	81.25	86.67	D093	39.58	45.42	49.58
D038	33.75	43.33	46.67	D094	87.08	93.33	97.08
D039	42.08	45.83	48.75	D095	69.58	79.58	90.42
D040	35	39.17	42.92	D096	60.42	69.17	75.42
D041	66.67	84.17	90	D097	28.75	39.17	47.5
D042	27.5	37.08	39.58	D098	62.92	83.33	92.08
D043	7.92	10	12.5	D099	52.92	74.17	88.33
D044	10	13.75	15	D100	17.08	26.67	31.67
D045	7.08	10	13.33	D101	64.17	78.75	89.58
D046	56.67	77.5	87.08	D102	86.25	99.17	99.58
D047	96.67	100	100	D103	71.67	99.17	100
D048	82.92	100	100	D104	79.17	99.17	100
D049	100	100	100	D105	64.17	97.92	99.58
D050	64.17	79.17	83.33	D106	44.58	92.5	99.17
D051	73.33	87.92	91.25	D107	23.33	41.25	53.75
D052	42.08	48.33	52.08	D108	17.92	26.25	31.67
D053	100	100	100	D109	41.67	53.75	63.33
D054	26.67	29.17	30.83	D110	46.67	70.83	81.67
D055	99.58	100	100	D111	46.25	60	67.08
D056	100	100	100	D112	36.67	47.08	48.33

Grey Level Co-occurrence Matrix

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	40	48.33	53.33
D002	24.17	31.25	34.58
D003	51.67	68.33	76.67
D004	59.17	78.33	92.08
D005	13.75	20	25.83
D006	57.08	66.25	70.42
D007	11.67	14.17	17.08
D008	62.5	81.25	90.83
D009	47.5	64.58	76.25
D010	43.75	49.17	51.67
D011	58.75	70.42	79.58
D012	44.58	47.08	47.5
D013	15	22.92	31.25
D014	65	77.5	85
D015	38.33	48.75	53.33
D016	100	100	100
D017	72.5	80.42	83.75
D018	41.25	55.42	62.08
D019	48.75	67.5	77.5
D020	75.83	84.17	92.92
D021	100	100	100
D022	16.25	28.33	32.92
D023	23.75	40.83	51.67
D024	68.33	85	94.17
D025	15	22.08	25.42
D026	60.83	72.5	79.17
D027	32.5	50.42	62.92
D028	40	52.08	57.92
D029	62.92	88.75	97.92
D030	13.33	25	33.75
D031	27.92	40.42	46.67
D032	30.83	39.17	43.75
D033	31.25	42.92	45
D034	41.67	51.25	58.75
D035	22.5	34.58	42.08
D036	23.33	35.42	44.58
D037	54.17	66.67	75.42
D038	20.83	27.92	31.25
D039	22.08	29.17	37.92
D040	18.33	25	29.17
D041	40.83	47.92	53.75
D042	13.33	15.42	18.75
D043	12.92	20.42	28.33
D044	6.67	12.5	16.25
D045	17.92	27.5	34.58
D046	42.5	52.5	63.33
D047	33.75	39.17	44.58
D048	35.83	40.42	46.67
D049	100	100	100
D050	14.58	26.67	31.25
D051	32.92	47.08	52.92
D052	24.17	29.58	34.58
D053	64.58	82.5	86.25
D054	11.67	18.33	22.92
D055	57.08	69.58	77.5
D056	68.75	86.67	93.33

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	52.92	73.75	83.75
D058	15.83	20.83	22.92
D059	7.92	10	14.17
D060	15.83	25.42	33.75
D061	15	28.75	35.42
D062	17.5	22.92	29.17
D063	13.75	18.75	23.75
D064	66.25	77.08	80.42
D065	90.42	98.75	99.17
D066	41.67	51.67	55.83
D067	39.17	56.67	68.75
D068	73.33	87.92	92.5
D069	41.25	56.67	63.33
D070	43.33	47.92	50
D071	62.92	83.75	91.25
D072	41.25	50.83	57.08
D073	19.58	30	36.67
D074	47.92	70.42	80
D075	31.25	37.08	44.58
D076	71.25	82.92	89.17
D077	79.17	90	92.08
D078	36.67	42.5	46.25
D079	77.5	91.25	94.58
D080	33.33	51.25	60
D081	31.67	46.25	54.17
D082	61.25	83.75	92.5
D083	41.25	47.5	57.08
D084	35.42	46.67	60.83
D085	31.25	38.75	45
D086	47.08	58.33	66.25
D087	50	67.5	77.08
D088	14.58	22.5	27.08
D089	12.92	20	23.75
D090	11.25	14.58	15.83
D091	10	17.5	21.67
D092	64.58	81.25	90
D093	25.42	32.92	36.67
D094	73.75	87.08	92.08
D095	35.83	44.58	51.25
D096	31.25	45	50.83
D097	22.08	35.83	44.17
D098	50.83	67.92	81.25
D099	26.25	39.58	50.42
D100	16.25	24.58	32.5
D101	62.08	77.08	88.33
D102	72.92	82.08	88.75
D103	92.08	99.58	100
D104	92.92	99.58	100
D105	68.75	89.58	92.5
D106	87.08	99.58	99.58
D107	41.25	55	62.5
D108	26.67	36.25	40
D109	38.33	51.67	59.58
D110	62.92	75	84.58
D111	26.25	37.08	47.5
D112	14.58	19.17	23.33

Multiresolution Simultaneous Autoregressive

Texture ID	Top matches considered			Texture ID	Top matches considered		
	Top 15	Top 30	Top 45		Top 15	Top 30	Top 45
D001	100	100	100	D057	99.17	100	100
D002	74.58	85	89.58	D058	34.58	40.83	52.92
D003	83.33	92.5	95.42	D059	15	28.75	37.08
D004	99.58	100	100	D060	65.42	81.25	89.58
D005	70	88.33	93.33	D061	66.25	80	85
D006	82.92	96.67	96.67	D062	55	73.33	84.17
D007	49.17	64.58	76.67	D063	40.42	59.58	73.75
D008	95	98.75	99.17	D064	99.17	100	100
D009	82.08	99.17	100	D065	98.75	100	100
D010	61.67	80	87.08	D066	96.67	100	100
D011	99.58	100	100	D067	52.5	63.33	67.08
D012	69.17	80.42	88.75	D068	97.92	100	100
D013	35	48.75	56.25	D069	24.17	27.08	31.67
D014	97.92	100	100	D070	99.17	100	100
D015	35.83	42.5	45	D071	40	53.75	61.67
D016	100	100	100	D072	54.58	75	85
D017	100	100	100	D073	45.83	59.58	66.25
D018	80.83	95.83	97.92	D074	77.5	92.5	95
D019	95.42	100	100	D075	83.33	94.58	99.58
D020	100	100	100	D076	98.75	100	100
D021	100	100	100	D077	100	100	100
D022	36.25	47.92	53.33	D078	97.08	99.58	100
D023	50	73.75	85.83	D079	96.25	100	100
D024	95.42	100	100	D080	95.42	100	100
D025	90.42	97.5	98.33	D081	89.58	98.75	100
D026	99.17	100	100	D082	99.17	100	100
D027	58.75	76.25	87.5	D083	100	100	100
D028	79.58	97.92	100	D084	99.17	100	100
D029	97.92	100	100	D085	95.42	99.58	100
D030	43.75	59.58	72.92	D086	89.17	97.08	98.33
D031	21.25	35.42	47.08	D087	81.25	95.83	99.58
D032	97.92	100	100	D088	24.58	41.67	50.42
D033	89.58	100	100	D089	25	50.42	59.17
D034	98.75	100	100	D090	46.25	54.58	62.08
D035	51.67	75.83	89.17	D091	54.58	65.42	80
D036	28.33	39.17	48.75	D092	68.75	86.25	94.17
D037	90.42	100	100	D093	61.25	67.92	70.42
D038	29.58	45	55	D094	100	100	100
D039	50	56.25	63.33	D095	100	100	100
D040	62.92	77.08	82.92	D096	76.67	90.42	94.58
D041	70.83	80.42	84.58	D097	23.75	34.17	43.75
D042	32.92	42.5	50	D098	50.42	73.33	87.08
D043	8.75	10.42	12.08	D099	27.08	42.92	53.33
D044	11.67	13.75	15.83	D100	48.75	65.42	72.92
D045	9.58	12.08	16.25	D101	84.58	97.92	99.58
D046	61.67	70.83	75.42	D102	84.17	99.17	100
D047	78.33	93.33	95.83	D103	67.92	99.17	100
D048	93.33	100	100	D104	72.5	100	100
D049	100	100	100	D105	49.17	72.08	78.75
D050	84.17	96.25	99.17	D106	39.58	60.83	76.67
D051	56.67	75.42	87.92	D107	39.17	52.92	64.58
D052	62.5	70.42	74.17	D108	27.5	35.83	45.42
D053	100	100	100	D109	41.67	69.17	87.5
D054	45	52.92	57.5	D110	50.83	83.33	96.67
D055	100	100	100	D111	69.58	83.33	90
D056	99.58	100	100	D112	53.75	69.58	78.75

Appendix B

Retrieval Rate of Brodatz Textures Using the Final DWF

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D001	88.33	97.92	100
D002	51.25	62.5	69.58
D003	51.25	58.33	62.08
D004	65	85.42	96.25
D005	65.83	82.92	87.5
D006	100	100	100
D007	69.17	84.17	90
D008	89.17	95	97.92
D009	69.17	87.5	95.42
D010	88.33	93.33	95.42
D011	100	100	100
D012	92.08	97.92	99.17
D013	38.75	55.42	60.42
D014	100	100	100
D015	55.42	66.25	70
D016	100	100	100
D017	100	100	100
D018	66.25	77.08	82.08
D019	99.17	100	100
D020	100	100	100
D021	100	100	100
D022	63.33	77.08	80
D023	44.17	69.17	82.92
D024	99.17	100	100
D025	74.58	83.33	87.92
D026	100	100	100
D027	55.83	77.08	89.17
D028	79.58	94.17	97.5
D029	98.75	100	100
D030	56.67	76.67	85.42
D031	64.17	84.58	88.33
D032	100	100	100
D033	97.08	100	100
D034	99.58	100	100
D035	92.92	96.25	97.92
D036	66.67	74.17	77.08
D037	67.5	87.92	92.5
D038	57.92	62.92	64.58
D039	78.75	91.25	93.33
D040	80.42	93.33	98.33
D041	83.33	93.33	95.42
D042	49.17	55	56.67
D043	8.75	12.08	13.75
D044	18.75	24.58	26.25
D045	23.75	32.5	43.75
D046	97.92	99.58	100
D047	100	100	100
D048	99.58	100	100
D049	100	100	100
D050	94.58	100	100
D051	99.58	100	100
D052	67.08	74.17	76.67
D053	100	100	100
D054	50.42	59.58	63.75
D055	89.58	95	96.67
D056	100	100	100

Texture ID	Top matches considered		
	Top 15	Top 30	Top 45
D057	99.17	100	100
D058	24.17	38.33	46.67
D059	32.92	42.08	47.5
D060	60.83	79.17	86.67
D061	75.42	92.92	96.67
D062	76.67	91.25	94.17
D063	54.58	72.92	80.42
D064	98.75	99.58	100
D065	99.58	100	100
D066	86.67	95.83	97.92
D067	60.42	67.5	70
D068	100	100	100
D069	35	42.08	47.08
D070	65.83	90	97.08
D071	72.08	89.17	95
D072	85	90.42	92.5
D073	45.83	57.92	64.17
D074	96.25	100	100
D075	99.58	100	100
D076	100	100	100
D077	100	100	100
D078	97.5	98.33	99.17
D079	91.67	96.25	97.92
D080	73.75	99.58	100
D081	88.33	99.17	100
D082	100	100	100
D083	100	100	100
D084	100	100	100
D085	97.5	99.58	100
D086	63.33	72.92	79.17
D087	93.33	100	100
D088	57.92	80	90.83
D089	47.92	67.08	78.75
D090	51.25	57.5	59.58
D091	45.83	57.5	69.17
D092	95	100	100
D093	75.42	87.92	92.08
D094	100	100	100
D095	84.58	100	100
D096	81.25	92.5	95.83
D097	27.92	37.92	42.08
D098	88.33	100	100
D099	42.92	70.83	88.75
D100	57.5	68.33	73.33
D101	77.08	99.58	100
D102	78.33	100	100
D103	67.92	98.75	100
D104	74.17	98.75	100
D105	56.67	95.83	100
D106	40	95.83	100
D107	82.08	97.5	99.17
D108	57.5	70.83	80.42
D109	59.17	94.58	99.58
D110	55.42	96.67	100
D111	87.92	96.67	98.75
D112	83.33	94.58	97.5

Appendix C

Classes of Vision Textures

The following textures from VisTex textures database are regarded as the same class. Each line represents a class of textures that can be barely discriminated by human vision.

Bark.0001	Bark.0001		
Bark.0009	Bark.0010		
Bark.0011	Bark.0012		
Brick.0000	Brick.0001		
Clouds.0000	Clouds.0001		
Fabric.0000	Fabric.0001		
Fabric.0002	Fabric.0003		
Fabric.0004	Fabric.0005		
Fabric.0008	Fabric.0010		
Fabric.0013	Fabric.0014		
Fabric.0015	Fabric.0016		
Fabric.0018	Fabric.0019		
Food.0002	Food.0004		
Food.0006	Food.0007	Food.0008	Food.0009
Grass.0001	Grass.0002		
Leaves.0006	Leaves.0007		
Metal.0000	Metal.0001		
Metal.0002	Metal.0003		
Metal.0004	Metal.0005		
Sand.0001	Sand.0002		
Sand.0003	Sand.0004		
Stone.0004	Stone.0005		
Tile.0007	Tile.0008	Tile.0009	Tile.0010
Water.0001	Water.0005		
Water.0003	Water.0006		
Waldo.0001	Waldo.0002		
Wood.0000	Wood.0001		
Paintings.4.0000	Paintings.4.0001		
All Terrain			

The following textures are highly nonhomogeneous and will not be used as texture query.

All Buildings
All Painting except Paintings.2.0001, Paintings.4.0000 and Paintings.4.0001
Stone.0000
Tile.0005
Tile.0006
Waldo.0000

Appendix D

Retrieval Rate of Vision Textures Using the Final DWF

Texture ID	Recall Rate
Bark.0000	43.75
Bark.0001	97.5
Bark.0002	95.42
Bark.0003	39.17
Bark.0004	62.5
Bark.0005	35.42
Bark.0006	52.5
Bark.0007	50.42
Bark.0008	42.5
Bark.0009	52.08
Bark.0010	34.17
Bark.0011	70
Bark.0012	48.33
Brick.0000	92.08
Brick.0001	90.83
Brick.0002	81.25
Brick.0003	53.75
Brick.0004	72.08
Brick.0005	57.5
Brick.0006	36.67
Brick.0007	16.25
Brick.0008	9.58
Clouds.0000	84.58
Clouds.0001	65.83
Fabrics.0000	68.33
Fabrics.0001	97.5
Fabrics.0002	79.17
Fabrics.0003	60
Fabrics.0004	68.33
Fabrics.0005	59.17
Fabrics.0006	61.25
Fabrics.0007	100
Fabrics.0008	73.33
Fabrics.0009	84.58
Fabrics.0010	86.67
Fabrics.0011	60.42
Fabrics.0012	54.17
Fabrics.0013	97.92
Fabrics.0014	100
Fabrics.0015	99.58
Fabrics.0016	98.33
Fabrics.0017	100
Fabrics.0018	99.58
Fabrics.0019	99.58
Flowers.0000	73.75
Flowers.0001	50
Flowers.0002	45
Flowers.0003	58.75

Texture ID	Recall Rate
Flowers.0004	59.17
Flowers.0005	58.33
Flowers.0006	86.67
Flowers.0007	52.08
Food.0000	82.5
Food.0001	99.17
Food.0002	74.58
Food.0003	81.25
Food.0004	50.42
Food.0005	79.58
Food.0006	99.17
Food.0007	100
Food.0008	100
Food.0009	100
Food.0010	52.08
Food.0011	96.67
Grass.0000	30.42
Grass.0001	97.92
Grass.0002	96.67
Leaves.0000	50.83
Leaves.0001	46.67
Leaves.0002	68.33
Leaves.0003	92.08
Leaves.0004	75.42
Leaves.0005	19.17
Leaves.0006	58.75
Leaves.0007	70.42
Leaves.0008	81.25
Leaves.0009	30.83
Leaves.0010	38.33
Leaves.0011	70
Leaves.0012	52.08
Leaves.0013	61.25
Leaves.0014	63.75
Leaves.0015	30.83
Leaves.0016	36.67
Metal.0000	93.75
Metal.0001	95.42
Metal.0002	100
Metal.0003	100
Metal.0004	91.67
Metal.0005	85
Misc.0000	76.25
Misc.0001	76.25
Misc.0002	73.33
Misc.0003	44.17
Paintings.2.0001	33.33
Paintings.4.0000	94.58

Texture ID	Recall Rate
Paintings.4.0001	53.33
Sand.0000	79.58
Sand.0001	100
Sand.0002	95.83
Sand.0003	96.25
Sand.0004	47.5
Sand.0005	40
Sand.0006	57.92
Stone.0008	55.83
Stone.0009	86.67
Stone.0010	36.25
Stone.0011	87.08
Stone.0012	85
Terrain.0000	72.5
Terrain.0001	67.5
Terrain.0002	59.58
Terrain.0003	85.83
Terrain.0004	79.17
Terrain.0005	74.58
Terrain.0006	90
Terrain.0007	90
Terrain.0008	47.92
Terrain.0009	92.08
Terrain.0010	75
Tile.0000	45
Tile.0001	32.08
Tile.0002	37.08
Tile.0003	41.67
Tile.0004	88.33
Tile.0007	100
Tile.0008	99.58
Tile.0009	100
Tile.0010	100
Water.0000	52.92
Water.0001	60
Water.0002	40
Water.0003	90
Water.0004	54.58
Water.0005	78.75
Water.0006	80
Water.0007	47.08
Waldo.0001	47.92
Waldo.0002	30
Wood.0000	27.5
Wood.0001	33.33
Wood.0002	86.25

Bibliography

[1] eVision Global, "A new vision for internet search - a technical white paper from evision," tech. rep., Available at http://www.evisionglobal.com, July 2001.

[2] Y. Rui and T. S. Huang, "Image retrieval: Current techniques, promising directions, and open issues," *Journal of Visual Communication and Image representation*, vol. 10, no. 1, pp. 39–62, 1999.

[3] A. W. M. Smeulders, M. L. Kersten, and T. Gevers, "Crossing the divide between computer vision and data bases in search of image databases," in *Proceedings Fourth Working Conference Visual Database Systems*, pp. 223–239, 1998.

[4] C. S. McCamy, H. Marcus, and J. G. Davidson, "A color rendition chart," *Journal of Applied Photographic Engineering*, vol. 2, no. 3, 1976.

[5] M. Miyahara, "Mathematical transform of (r,g,b) color data to mulsel (h,s,v) color data," *SPIE Visual Communication Image Processing*, vol. 1001, 1988.

[6] J. Wang, W.-J. Yang, and R. Acharya, "Color clustering techniques for color-content-based image retrieval from image databases," in *Proceeding IEEE Conference on Multimedia Computing and Systems*, 1997.

[7] M. Swain and D. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, 1991.

[8] M. Ioka, "A method of determing the similarity of images on the basis of color information," Tech. Rep. RT-0030, IBM Research, Tokyo Research Laboratory, November 1989.

[9] W. Niblack, R. Barber, and et al., "The qbic project: Querying images by content using color, texture and shape," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, February 1994.

[10] M. Stricker and M. Orengo, "Similarity of color images," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, 1995.

[11] J. R. Smith and S.-F. Chang, "Single color extraction and image query," in *Proceeding IEEE International Conference on Image Processing*, 1995.

[12] G. Pass, R. Zabeh, and J. Miller, "Comparing images using color coherence vector," *ACM Multimedia*, pp. 65–73, 1996.

[13] Y. Rui, A. C. She, and T. S. Huang, "Modified fourier descriptors for shape representation-a practical approach," in *Proceeding First International Workshop on Image Databases and Multimedia Research*, 1996.

[14] E. Person and K. S. Fu, "Shape discrimination using fourier descriptors," *IEEE Transaction on System, Man and Cybernetics*, 1977.

[15] M. K. Hu, "Visual pattern recognition by moments invariants, computer methods in image analysis," *IRE Transactions on Information Theory*, vol. 8, 1962.

[16] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America*, vol. 70, no. 8, pp. 920–930, 1979.

[17] A. Pentland, R. W. Pickard, and S. Sclaroff, "Photobook: Content-based manipulation of image databases," *International Journal of Computer Vision*, 1996.

[18] E. M. Arkin, L. Chew, D. Huttenlocher, K. Kedem, and J. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transaction on Pattern Recognition and Machine Intelligence*, vol. 13, no. 3, 1991.

[19] C. C.-H. Chuang and C.-C. J. Kuo, "Wavelet descriptor of planar curves: Theory and applications," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 56–70, 1996.

[20] D. White and R. Jain, "Similarity indexing: Algorithm and performance," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, 1996.

[21] A. Guttman, "R-tree: A dynamic index structure for spatial searching," in *Proceeding ACM SIGMOD*, 1984.

[22] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The r+-tree: A dynamic index for multi-dimensional objects," in *Proceeding 12th VLDB*, 1987.

[23] D. Greene, "An implementation and performance analysis of spatial data access," in *Proceeding ACM SIGMOD*, 1989.

[24] N. Beckman, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *Proceeding ACM SIGMOD*, 1990.

[25] Y. Rui, K. Chakrabarti, S. Mehrotra, Y. Zhao, and T. S. Huang, "Dynamic clustering for optimal retrieval in high dimensional multimedia databases," *TR-MARS-10-97*, 1997.

[26] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[27] H. J. Zhang and D. Zhong, "A scheme for visual feature based image retrieval," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, 1995.

[28] V. Gaede and O. Gunther, "Multidimensional access methods," Tech. Rep. 16, Institut fur Wirtschaftsinformatik Humboldt-Universitat zu Berlin, August 1995. Information System Series.

[29] C. Faloutsos, M. Flickner, W. Niblack, D. Petkovic, W. Equitz, and R. Barber, "Efficient and effective querying by image content," tech. rep., IBM Research Report, 1993.

[30] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Proceeding 4th International Conference on Foundations of Data Organization and Algorithm*, pp. 69–84, 1993.

[31] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualisation of traditional and multimedia datasets," Tech. Rep. CS-TR-3383, Dept. of Computer Science, University of Maryland at College Park, January 1995.

[32] R. Ng and A. Sedighian, "Evaluating multidimensional indexing structures for images transformed by principal component analysis," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, 1996.

[33] C. Faloutsos and K.-I. Lin, "Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *Proceeding SIGMOD*, pp. 163–174, 1995.

[34] S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, and et al., "An eigenspace update algorithm for image analysis," *CVGIP:Graphical Models and Image Processing*, 1997.

[35] G. Salton and M. J. McGill, *Introduction to modern information retrieval*. McGraw-Hill, NewYork, 1983.

[36] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[37] J. Eakins and M. Graham, "Content-based image retrieval: A report to the jisc technology applications programme," Tech. Rep. JTAP-039, Institute for Image and Data Research, University of Northumbria at Newcastle, January 1999.

[38] C. Venters and M. Cooper, "Content-based image retrieval," Tech. Rep. JTAP-054, JISC Technology Application Program, 2000.

[39] M. Flickner, H. Sawhney, W. Niblack, and et al., "Query by image and video content: The qbic system," *IEEE Computer*, vol. 28, pp. 23–32, September 1995.

[40] W. Niblack, Z. Xiaoming, J. L. Hafner, and et al., "Updates to the qbic system," in *Proceeding SPIE Storage and Retrieval for Image and Video databases*, pp. 150–161, 1998.

[41] H. Tamura, S. Mori, and T. Yamawaki, "Textural features corresponding to visual perception," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-8, pp. 460–473, June 1978.

[42] J. R. Bach, C. Fuller, A. Gupta, and et al., "Virage image search engine: An open framework for image management," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, pp. 76–87, 1996.

[43] A. Hampapur, A. Gupta, B. Horrowitz, and et al., "Virage video engine," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, pp. 188–197, 1997.

[44] J. Dowe, "Content-based retrieval in multimedia imaging," in *Proceeding SPIE Storage and Retrieval for Image and Video Databases*, 1993.

[45] RetrievalWare, "Demo page." http://vrw/excalib.com/cgi-bin/sdk/cst/cst2.bat, 1997.

[46] J. R. Smith and S.-F. Chang, "Visualseek: A fully automated content-based image query system," in *Proceeding ACM Multimedia*, pp. 87–98, 1996.

[47] W. Y. Ma and B. S. Manjunath, "Netra: A toolbox for navigating large image databases," in *Proceeding IEEE International Conference on Image Processing*, 1997.

[48] T. S. Huang, S. Mehrotra, and K. Ramachandran, "Multimedia analysis and retrieval system (mars) project," in *Proceeding 33rd Annual Clinic on Library Application of Data Processing-Digital Image Access and Retrieval*, 1996.

[49] T. Gevers and A. W. M. Smeulders, "Pictoseek: A color invariant retrieval system," in *Image Databases and Multimedia Search*, pp. 25–37, 1997.

[50] I. K. Sethi, I. Coman, B. Day, and et al., "Color-wise: A system for image similarity retrieval using color," in *Proceeding SPIE Storage and Retrieval for Image and Video databases*, pp. 140–149, 1998.

[51] D. Forsyth, J. Malik, and R. Wilensky, "Searching for digital pictures," in *Scientific American*, pp. 72–77, July 1997.

[52] P. Alsuth, T. Hermes, and J. Kreyss, "Image-miner - intelligent retrieval for images and videos," in *Image Databases and Multimedia Search*, pp. 241–251, 1997.

[53] K. Hirata and T. Kato, "Query by visual example," in *Proceeding 3rd International Conference on Extending Database Technology*, 1992.

[54] H. H. Yu and W. Wolf, "Hierarchical, multiresolution algorithm for dictionary-driven cbir," in *Proceeding IEEE International Conference on Image Processing*, 1997.

[55] "Imatch by mwlabs." http://www.photools.com, http://www.mwlabs.de.

[56] "Dart by at&t." http://www.uk.research.att.com/dart.

[57] W. Wang, Y. Song, and A. Zhang, "Semantics-based image retrieval by region saliency," in *Proceeding International Conference on Image and Video Retrieval (LNCS 2382)*, pp. 29–37, 2002.

[58] K. Barnard, "Recognition as translating images into text," in *Proceeding SPIE Internet Imaging IV*, 2003.

[59] K. Kim, J. Choi, N. Kim, and P. Kim, "Extracting semantic information from basketball video based on audio-visual features," in *Proceeding International Conference on Image and Video Retrieval (LNCS 2382)*, pp. 278–288, 2002.

[60] P. Rubens, "Fax - the technology that refuses to die." http://news.bbc.co.uk/2/hi/uk_news/magazine/3320515.stm.

[61] J. M. Coggins, *A framework for texture analysis based on spatial filtering.* PhD thesis, Computer Science Department, Michigan State University, 1982.

[62] J. Sklansky, "Image segmentation and feature extraction," *IEEE Transactions on Systems. man and Cybernetics*, vol. SMC-8, pp. 237–247, 1978.

[63] R. M. Haralick, "Statistical and structural approach to texture," *Proceeding of the IEEE*, vol. 67, pp. 786–804, 1979.

[64] J. K. Hawkins, "Textural properties for pattern recognition," *Picture Processing and Psychopictorics*, 1969. Academic Press.

[65] M. Tuceryan and A. K. Jain, "Texture analysis," *The Handbook of Pattern Recognition and Computer Vision (2nd Edition)*, pp. 207–248, 1998. World Scientific Publishing Co.

[66] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 610–621, November 1973.

[67] B. S. Manjunath and R. Chellappa, "Unsupervised texture segmentation using markov random fields models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 478–482, May 1991.

[68] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian markov random fields," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-33, pp. 959–963, August 1985.

[69] R. L. Kashyap and R. Chellappa, "Estimation and choice of neighbors in spatial-interaction models of images," *IEEE Transactions on Information theory*, vol. 29, pp. 60–72, January 1983.

[70] J. C. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, no. 2, pp. 173–188, 1992.

[71] A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-6, pp. 661–674, November 1984.

[72] L. M. Kaplan, "Extended fractal analysis for texture classification and segmentation," *IEEE Transactions on Image Processing*, vol. 8, pp. 1572–1585, November 1999.

[73] B. B. Chaudhuri and N. Sarkar, "Texture segmentation using fractal dimension," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 72–77, January 1995.

[74] D. Dunn and W. E. Higgins, "Optimal gabor filters for texture segmentation," *IEEE Transactions on Image Processing*, vol. 4, pp. 947–964, July 1995.

[75] T. P. Weldon and W. E. Higgins, "Design of multiple gabor filters for texture segmentation," in *IEEE International Conference on Accoustics, Speech and Signal Processing*, vol. 4, pp. 2243–2246, 1996.

[76] J. R. Smith and S.-F. Chang, "Transform features for texture classification and discrimination in large image databases," in *Proceedings of IEEE International Conference on Image Processing (ICIP '94)*, vol. 3, pp. 407–411, 1994.

[77] T. Chang and C. J. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Transactions on Image Processing*, vol. 2, pp. 429–441, October 1993.

[78] M. Unser, "Texture classification and segmentation using wavelet frames," *IEEE Transactions on Image Processing*, vol. 4, pp. 1549–1560, November 1995.

[79] K. S. Thygarajan, T. Nguyen, and C. E. Persons, "A maximum likelihood approach to texture classification using wavelet transform," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 64–644, 1994.

[80] A. Kundu and J.-L. Chen, "Texture classification using qmf bank-based subband decomposition," *CVGIP: Graphical Models and Image Processing*, vol. 54, pp. 369–384, September 1992.

[81] V. Manian, R. Vasquez, and P. Katiyar, "Texture classification using logical operators," *IEEE Transaction on Image Processing*, vol. 9, pp. 1693–1703, October 2000.

[82] D. Coltuc, J.-M. Becker, and V. Buzuloiu, "Jordan features for texture segmentation," in *Proceedings of the 3rd IEEE International Conference on Electronic Circuits and Systems*, vol. 1, pp. 195–198, 1996.

[83] Y. Q. Chen, M. S. Nixon, and D. W. Thomas, "Texture classification using statistical geometrical features," in *Proceedings of IEEE International Conference on Image Processing*, vol. 3, pp. 446–450, 1994.

[84] C.-M. Wu, Y.-C. Chen, and K. S. Hsieh, "Texture features for classification of ultrasonic liver images," *IEEE Transactions on Medical Imaging*, vol. 11, pp. 141–152, June 1992.

[85] K. I. Kim, K. Jung, S. H. Park, and H. J. Kim, "Texture classification with kernel principal component analysi," *Electronics Letters*, vol. 36, pp. 1021–1022, June 2000.

[86] N. Paragios and R. Deriche, "Geodasic active contours for supervised texture segmentation," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 422–427, 1999.

[87] M. Tuceryan, A. K. Jain, and Y. Lee, "Texture segmentation using voronoi polygon," in *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 94–99, 1988.

[88] J. Kuan, *Image texture analysis with fast similarity search for content based retrieval and navigation.* Phd thesis, University of Southampton, 1998.

[89] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, pp. 269–285, April 1976.

[90] P. P. Ohanian and R. C. Dubes, "Performance evaluation for four classes of textural features," *Pattern Recognition*, vol. 25, no. 8, pp. 819–833, 1992.

[91] W. Y. Ma and B. S. Manjunath, "A comparison of wavelet transform features for texture image annotation," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, pp. 256–259, 1995.

[92] N. Ahmed and K. R. Rao, *Orthogonal transform for digital signal processing.* Springer, NewYork, 1975.

[93] S. G. Mallat, "A theory for multiresolution signal decomposition: The wavelet representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 674–693, July 1989.

[94] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, pp. 961–1005, September 1990.

[95] A. Graps, "An introduction to wavelets," in *IEEE Computational Science and Engineering*, vol. 2, pp. 50–61, 1995.

[96] "Wavelet toolbox for matlab version 6.1."

[97] R. C. Gonzales and R. E. Woods, *Digital Image Processing.* Addison-Wesley, 3rd ed., 1992.

[98] M. Nachtegael, D. V. der Weken, D. V. D. Ville, E. Kerre, W. Philips, and I. Lemahieu, "A comparative study of classical and fuzzy filters for noise reduction," in *Proceedings of The 10th IEEE International Conference on Fuzzy Systems*, pp. 11–14, 2001.

[99] R. A. Peters, "A new algorithm for image noise reduction using mathematical morphology," *IEEE Transactions on Image Processing*, vol. 4, pp. 554–568, May 1995.

[100] M. Yoshioka and S. Omatu, "Noise reduction method for image processing using genetic algorithm," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2650–2655, 1997.

[101] Q. Zhang, P. A. Mlsna, and J. J. Rodriguez, "A recursive technique for 3-d histogram enhancement of color images," in *Proceedings of IEEE Southwest Symposium on Image Analysis and Interpretation*, pp. 218–223, 1996.

[102] H.-S. Wong and J.-H. Wang, "Contrast enhancement based on divided histogram manipulation," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1551–1555, 2000.

[103] M. F. A. Fauzi and P. H. Lewis, "Query by fax for content-based image retrieval," in *Proceedings of International Conference on the Challenge of Image and Video Retrieval (Springer's Lecture Notes in Computer Science vol. 2383)*, pp. 91–99, 2002.

[104] P. H. Lewis, K. Martinez, F. S. Abas, M. F. A. Fauzi, and et al., "An integrated content and metadata-based retrieval system for art," *IEEE Transactions on Image Processing*, vol. 13, pp. 302–313, March 2004.

[105] M. D. Swanson and A. H. Tewfik, "A binary wavelet decomposition of binary images," *IEEE Transactions on Image Processing*, vol. 5, pp. 1637–1650, December 1996.

[106] L. S. Davis, S. A. John, and J. K. Agrawal, "Texture analysis using generalized co-occurrence matrix," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 251–259, July 1979.

[107] G. R. Cross and A. K. Jain, "Markov random field texture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 25–39, January 1983.

[108] B. B. Mandelbrot and J. V. Ness, "Fractional brownian motion, fractional noise and applications," *SIAM Review*, vol. 10, 1968.

[109] J. M. Keller, S. Chen, and R. M. Crownover, "Texture description and segmentation through fractal geometry," *Computer Vision, Graphics, and Image Processing*, vol. 45, no. 2, pp. 150–166, 1989.

[110] B. B. Chaudhuri and N. Sarkar, "An efficient approach to estimate fractal dimension of textural images," *Pattern Recognition*, vol. 25, no. 9, pp. 1035–1041, 1992.

[111] K. I. Laws, *Textured Image Segmentation.* Phd thesis, University of Southern California, 1980.

[112] T. A. Ramstad, S. O. Aase, and J. H. Husoy, *Subband Compression of Images - Principles and Examples.* ELSEVIER Science Publishers, 1995.

[113] I. Ng, T. Tan, and J. Kittler, "On local linear transform and gabor filter representation of texture," in *Proceeding International Conference on Pattern Recognition*, pp. 627–631, 1992.

[114] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 837–842, August 1996.

[115] A. Mojsilovic, S. Markovic, and M. Popovic, "Texture analysis and classification with the nonseparable wavelet transform," in *Proceedings of International Conference on Image Processing*, vol. 3, pp. 182–185, 1997.

[116] P. de Rivaz and N. Kingsbury, "Complex wavelet features for fast texture image retrieval," in *Proceeding of International Conference on Image Processing (ICIP '98)*, vol. 1, pp. 109–113, 1999.

[117] N.-D. Kim and S. Udpa, "Texture classification using rotated wavelet filters," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 30, pp. 847–852, November 2000.

[118] A. Laine and J. Fan, "Texture classification by wavelet packet signatures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 1186–1191, November 1993.

[119] T.-H. Chang, Y.-C. Lin, and C. C. J. Kuo, "Techniques in texture analysis," *Medical Imaging Systems Techniques and Applications: Computational Techniques*, pp. 207–248, 1998. Gordon and Breach Science Publishers.

[120] S. Mallat, "Zero crossings of a wavelet transform," *IEEE Transactions on Information Theory*, vol. 37, July 1991.

[121] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, July 1992.

[122] A. Rosenfeld and S. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Transactions on Computers*, vol. C-20, 1971.

[123] R. Picard, C. Graczyk, S. Mann, and et al., "Vision texture 1.0," tech. rep., Media Laboratory, MIT, 1995. http://www-white.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

[124] J. R. Smith and S.-F. Chang, "Quad-tree segmentation for texture based image query," in *Proceedings of the 2nd Annual ACM Multimedia Conference*, (San Francisco), 1994.

[125] A. Natsev, R. Rastogi, and K. Shim, "Walrus: A similarity retrieval algorithm for image databases," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 395–406, 1999.

[126] J. Guo and A. Zhang, "Image decomposition and representation in large image database systems," *Journal of Visual Communication and Image Representation*, vol. 8, pp. 167–181, June 1997.

[127] S. Chan, K. Martinez, P. Lewis, C. Lahanier, and J. Stevenson, "Handling sub-image queries in content-based retrieval of high resolution art images," in *Proceedings of International Conference in Cultural Heritage and Technologies*, pp. 157–163, 2001.

[128] M. F. A. Fauzi and P. H. Lewis, "Texture-based image retrieval using multiscale sub-image matching," in *Proceedings of IST/SPIE Symposium on Electronic Imaging: Image and Video Communications and Processing 2003 (SPIE vol. 5022)*, pp. 407–416, 2003.

[129] S. N. Talbar, R. S. Holambe, and T. R. Sontakke, "Supervised texture classification using wavelet transform," in *Proceedings of the 4th International Conference on Signal Processing (ICSP '98)*, vol. 2, pp. 1177–1180, 1998.

[130] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, 1991.

[131] M. Tuceryan and A. K. Jain, "Texture segmentation using voronoi polygon," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 211–216, February 1990.

[132] I. Ng, J. Kittler, and J. Illingworth, "Supervised segmentation using a multiresolution data representation," *Signal Processing*, vol. 3, p. March, 133–163 1993.

[133] E. Salari and Z. Ling, "Texture segmentation using hierarchical wavelet decomposition," in *Proceedings of the IEEE International Symposium on Industrial Electronics*, vol. 1, pp. 216–220, 1995.

[134] T. Chang and C. C. J. Kuo, "Texture segmentation with tree-structured wavelet transform," in *Proceedings of the IEEE-SP International Symposium on Tine-Frequency and Time-Scale Analysis*, pp. 543–546, 1992.

[135] S. Krishnamachari and R. Chellappa, "Multiresolution gauss-markov random field models for texture segmentation," *IEEE Transactions on Image Processing*, vol. 6, pp. 251–267, February 1997.

[136] A. Perry and D. G. Lowe, "Segmentation of textured images," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '89)*, pp. 319–325, 1989.

[137] R. Porter and N. Canagarajah, "A robust automatic clustering scheme for image segmentation using wavelets," *IEEE Transactions on Image Processing*, vol. 5, pp. 662–665, April 1996.

[138] J. D. Buf, M. Kardan, and M. Spann, "Texture feature performance for image segmentation," *Pattern recognition*, vol. 23, no. 3/4, pp. 291–309, 1990.

[139] K. I. Chang, K. W. Bowyer, and M. Sivagurunath, "Evaluation of texture segmentation algorithms," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 294–299, 1999.

[140] O. Pichler, A. Teuner, and B. J. Hosticka, "A comparison of texture feature extraction using adaptive gabor filtering, pyramidal and tree structured wavelet transform," *Pattern Recognition*, vol. 29, no. 5, pp. 733–742, 1996.

[141] M. F. A. Fauzi and P. H. Lewis, "A fully unsupervised texture segmentation algorithm," in *Proceedings of British Machine Vision Conference 2003*, pp. 519–528, 2003.

[142] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[143] D. Comaniciu and P. Meer, "Distribution free decomposition of multivariate data," *Pattern Analysis and Applications*, vol. 2, pp. 22–30, 1999.

[144] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of International Conference on Computer Vision*, pp. 1197–1203, 1999.

[145] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 32–40, January 1975.

[146] "Artiste project." http://www.artisteweb.org.

[147] "Sculpteur project." http://www.sculpteurweb.org.